

Description of STM32L1 HAL and low-layer drivers

Introduction

STMCube™ is an STMicroelectronics original initiative to make developers' lives easier by reducing development effort, time and cost. STM32Cube covers the whole STM32 portfolio.

STM32Cube Version 1.x includes:

- The STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per Series (such as STM32CubeL1 for STM32L1 Series)
 - The STM32Cube HAL, STM32 abstraction layer embedded software ensuring maximized portability across the STM32 portfolio.
 - The low-layer APIs (LL) offering a fast light-weight expert-oriented layer which is closer to the hardware than the HAL. LL APIs are available only for a set of peripherals.
 - A consistent set of middleware components such as RTOS, USB, TCP/IP and Graphics.
 - All embedded software delivered with a full set of examples.

The HAL driver layer provides a generic multi-instance simple set of APIs (application programming interfaces) to interact with the upper layer (application, libraries and stacks).

The HAL driver APIs are split into two categories: generic APIs which provide common and generic functions for all the STM32 series and extension APIs which include specific and customized functions for a given line or part number. The HAL drivers include a complete set of ready-to-use APIs which simplify the user application implementation. As an example, the communication peripherals contain APIs to initialize and configure the peripheral, manage data transfers in polling mode, handle interrupts or DMA, and manage communication errors.

The HAL drivers are feature-oriented instead of IP-oriented. As an example, the timer APIs are split into several categories following the IP functions: basic timer, capture, pulse width modulation (PWM), and so on. The HAL driver layer implements run-time failure detection by checking the input values of all functions. Such dynamic checking contributes to enhance the firmware robustness. Run-time detection is also suitable for user application development and debugging.

The LL drivers offer hardware services based on the available features of the STM32 peripherals. These services reflect exactly the hardware capabilities and provide atomic operations that must be called following the programming model described in the product line reference manual. As a result, the LL services are not based on standalone processes and do not require any additional memory resources to save their states, counter or data pointers: all operations are performed by changing the associated peripheral registers content. Contrary to the HAL, the LL APIs are not provided for peripherals for which optimized access is not a key feature, or for those requiring heavy software configuration and/or complex upper level stack (such as USB).

The HAL and LL are complementary and cover a wide range of applications requirements:

- The HAL offers high-level and feature-oriented APIs, with a high-portability level. They hide the MCU and peripheral complexity to end-user.
- The LL offers low-level APIs at registers level, with better optimization but less portability. They require deep knowledge of the MCU and peripherals specifications.

The source code of HAL and LL drivers is developed in Strict ANSI-C which makes it independent from the development tools. It is checked with CodeSonar™ static analysis tool. It is fully documented and is MISRA-C 2004 compliant.



Contents

1 Acronyms and definitions.....	24
2 Overview of HAL drivers	26
2.1 HAL and user-application files.....	26
2.1.1 HAL driver files	26
2.1.2 User-application files	27
2.2 HAL data structures	29
2.2.1 Peripheral handle structures	29
2.2.2 Initialization and configuration structure	30
2.2.3 Specific process structures	30
2.3 API classification	30
2.4 Devices supported by HAL drivers	31
2.5 HAL driver rules	35
2.5.1 HAL API naming rules	35
2.5.2 HAL general naming rules	36
2.5.3 HAL interrupt handler and callback functions.....	37
2.6 HAL generic APIs.....	38
2.7 HAL extension APIs	39
2.7.1 HAL extension model overview	39
2.7.2 HAL extension model cases	39
2.8 File inclusion model.....	41
2.9 HAL common resources.....	42
2.10 HAL configuration.....	43
2.11 HAL system peripheral handling	44
2.11.1 Clock.....	44
2.11.2 GPIOs.....	45
2.11.3 Cortex NVIC and SysTick timer.....	46
2.11.4 PWR	46
2.11.5 EXTI.....	47
2.11.6 DMA.....	48
2.12 How to use HAL drivers	49
2.12.1 HAL usage models	49
2.12.2 HAL initialization	50
2.12.3 HAL IO operation process	52
2.12.4 Timeout and error management.....	55
3 Overview of Low Layer drivers.....	59

3.1	Low Layer files	59
3.2	Overview of Low Layer APIs and naming rules.....	61
3.2.1	Peripheral initialization functions	61
3.2.2	Peripheral register-level configuration functions	64
4	Cohabiting of HAL and LL	66
4.1	Low Layer driver used in standalone mode.....	66
4.2	Mixed use of Low Layer APIs and HAL drivers	66
5	HAL System Driver	67
5.1	HAL Firmware driver API description	67
5.1.1	How to use this driver	67
5.1.2	Initialization and de-initialization functions	67
5.1.3	HAL Control functions.....	67
5.1.4	Detailed description of functions	68
5.2	HAL Firmware driver defines.....	71
5.2.1	HAL.....	71
6	HAL ADC Generic Driver.....	74
6.1	ADC Firmware driver registers structures	74
6.1.1	ADC_InitTypeDef.....	74
6.1.2	ADC_ChannelConfTypeDef	76
6.1.3	ADC_AnalogWDGConfTypeDef.....	77
6.1.4	ADC_HandleTypeDef	77
6.2	ADC Firmware driver API description.....	78
6.2.1	ADC peripheral features	78
6.2.2	How to use this driver	78
6.2.3	Initialization and de-initialization functions	81
6.2.4	IO operation functions	81
6.2.5	Peripheral Control functions	82
6.2.6	Peripheral State and Errors functions	82
6.2.7	Detailed description of functions	82
6.3	ADC Firmware driver defines	88
6.3.1	ADC	88
7	HAL ADC Extension Driver	98
7.1	ADCEx Firmware driver registers structures	98
7.1.1	ADC_InjectionConfTypeDef	98
7.2	ADCEx Firmware driver API description	99
7.2.1	IO operation functions	99

Contents	UM1816
7.2.2 Peripheral Control functions	100
7.2.3 Detailed description of functions	100
7.3 ADCEx Firmware driver defines	102
7.3.1 ADCEx	102
8 HAL COMP Generic Driver.....	105
8.1 COMP Firmware driver registers structures	105
8.1.1 COMP_InitTypeDef	105
8.1.2 COMP_HandleTypeDef	106
8.2 COMP Firmware driver API description	106
8.2.1 COMP Peripheral features	106
8.2.2 How to use this driver	106
8.2.3 Initialization and de-initialization functions	107
8.2.4 IO operation functions	107
8.2.5 Peripheral Control functions	107
8.2.6 Peripheral State functions	108
8.2.7 Detailed description of functions	108
8.3 COMP Firmware driver defines	110
8.3.1 COMP	110
9 HAL COMP Extension Driver.....	117
9.1 COMPEx Firmware driver defines	117
9.1.1 COMPEx.....	117
10 HAL CORTEX Generic Driver.....	119
10.1 CORTEX Firmware driver registers structures	119
10.1.1 MPU_Region_InitTypeDef.....	119
10.2 CORTEX Firmware driver API description	120
10.2.1 Initialization and de-initialization functions	120
10.2.2 Peripheral Control functions	120
10.2.3 Detailed description of functions	120
10.3 CORTEX Firmware driver defines	125
10.3.1 CORTEX.....	125
11 HAL CRC Generic Driver.....	128
11.1 CRC Firmware driver registers structures	128
11.1.1 CRC_HandleTypeDef.....	128
11.2 CRC Firmware driver API description	128
11.2.1 How to use this driver	128
11.2.2 Initialization and de-initialization functions	128

11.2.3	Peripheral Control functions	128
11.2.4	Peripheral State functions	129
11.2.5	Detailed description of functions	129
11.3	CRC Firmware driver defines	130
11.3.1	CRC	130
12	HAL CRYP Generic Driver.....	132
12.1	CRYP Firmware driver registers structures	132
12.1.1	CRYP_InitTypeDef	132
12.1.2	CRYP_HandleTypeDef.....	132
12.2	CRYP Firmware driver API description	133
12.2.1	Initialization and de-initialization functions	133
12.2.2	AES processing functions	133
12.2.3	DMA callback functions	134
12.2.4	CRYP IRQ handler management.....	134
12.2.5	Peripheral State functions	134
12.2.6	Detailed description of functions	134
12.3	CRYP Firmware driver defines.....	142
12.3.1	CRYP	142
13	HAL CRYP Extension Driver.....	146
13.1	CRYPEX Firmware driver API description	146
13.1.1	Extended features functions	146
13.1.2	Detailed description of functions	146
14	HAL DAC Generic Driver.....	147
14.1	DAC Firmware driver registers structures	147
14.1.1	DAC_HandleTypeDef	147
14.1.2	DAC_ChannelConfTypeDef	147
14.2	DAC Firmware driver API description.....	147
14.2.1	DAC Peripheral features.....	147
14.2.2	How to use this driver.....	149
14.2.3	Initialization and de-initialization functions	150
14.2.4	IO operation functions	150
14.2.5	Peripheral Control functions	151
14.2.6	Peripheral State and Errors functions	151
14.2.7	Detailed description of functions	151
14.3	DAC Firmware driver defines	155
14.3.1	DAC	155

15 HAL DAC Extension Driver	159
15.1 DACEEx Firmware driver API description	159
15.1.1 How to use this driver	159
15.1.2 Extended features functions	159
15.1.3 Detailed description of functions	159
15.2 DACEEx Firmware driver defines	163
15.2.1 DACEEx	163
16 HAL DMA Generic Driver	165
16.1 DMA Firmware driver registers structures	165
16.1.1 DMA_InitTypeDef	165
16.1.2 __DMA_HandleTypeDef	165
16.2 DMA Firmware driver API description	166
16.2.1 How to use this driver	166
16.2.2 Initialization and de-initialization functions	167
16.2.3 IO operation functions	167
16.2.4 Peripheral State and Errors functions	168
16.2.5 Detailed description of functions	168
16.3 DMA Firmware driver defines	171
16.3.1 DMA	171
17 HAL FLASH Generic Driver.....	177
17.1 FLASH Firmware driver registers structures	177
17.1.1 FLASH_ProcessTypeDef	177
17.2 FLASH Firmware driver API description.....	177
17.2.1 FLASH peripheral features	177
17.2.2 How to use this driver	177
17.2.3 Programming operation functions	178
17.2.4 Option Bytes Programming functions	179
17.2.5 Peripheral Control functions	179
17.2.6 Peripheral Errors functions	179
17.2.7 Detailed description of functions	180
17.3 FLASH Firmware driver defines	182
17.3.1 FLASH	182
18 HAL FLASH Extension Driver	185
18.1 FLASHEEx Firmware driver registers structures	185
18.1.1 FLASH_EraseInitTypeDef	185
18.1.2 FLASH_OBProgramInitTypeDef	185

18.1.3	FLASH_AdvOBProgramInitTypeDef	186
18.2	FLASHEx Firmware driver API description.....	186
18.2.1	FLASH Erasing Programming functions.....	186
18.2.2	Option Bytes Programming functions.....	186
18.2.3	DATA EEPROM Programming functions	187
18.2.4	Detailed description of functions	187
18.3	FLASHEx Firmware driver defines	191
18.3.1	FLASHEx	191
19	HAL FLASH__RAMFUNC Generic Driver.....	198
19.1	FLASH__RAMFUNC Firmware driver API description.....	198
19.1.1	Peripheral errors functions	198
19.1.2	Detailed description of functions	198
20	HAL GPIO Generic Driver.....	202
20.1	GPIO Firmware driver registers structures	202
20.1.1	GPIO_InitTypeDef	202
20.2	GPIO Firmware driver API description	202
20.2.1	GPIO Peripheral features	202
20.2.2	How to use this driver	203
20.2.3	Initialization and Configuration functions.....	203
20.2.4	Detailed description of functions	204
20.3	GPIO Firmware driver defines.....	206
20.3.1	GPIO.....	206
21	HAL GPIO Extension Driver	209
21.1	GPIOEx Firmware driver defines.....	209
21.1.1	GPIOEx	209
22	HAL I2C Generic Driver	210
22.1	I2C Firmware driver registers structures	210
22.1.1	I2C_InitTypeDef.....	210
22.1.2	I2C_HandleTypeDef	210
22.2	I2C Firmware driver API description.....	211
22.2.1	How to use this driver	211
22.2.2	Initialization and de-initialization functions	216
22.2.3	IO operation functions	216
22.2.4	Peripheral State, Mode and Error functions	218
22.2.5	Detailed description of functions	218
22.3	I2C Firmware driver defines	230

Contents	UM1816
22.3.1 I2C	230
23 HAL I2S Generic Driver	236
23.1 I2S Firmware driver registers structures	236
23.1.1 I2S_InitTypeDef.....	236
23.1.2 I2S_HandleTypeDef	236
23.2 I2S Firmware driver API description	237
23.2.1 How to use this driver	237
23.2.2 Initialization and de-initialization functions	239
23.2.3 IO operation functions	239
23.2.4 Peripheral State and Errors functions	240
23.2.5 Detailed description of functions	240
23.3 I2S Firmware driver defines	245
23.3.1 I2S	245
24 HAL IRDA Generic Driver	250
24.1 IRDA Firmware driver registers structures	250
24.1.1 IRDA_InitTypeDef.....	250
24.1.2 IRDA_HandleTypeDef	250
24.2 IRDA Firmware driver API description	251
24.2.1 How to use this driver	251
24.2.2 Initialization and Configuration functions	253
24.2.3 IO operation functions	253
24.2.4 Peripheral State and Errors functions	254
24.2.5 Detailed description of functions	254
24.3 IRDA Firmware driver defines	259
24.3.1 IRDA	259
25 HAL IWDG Generic Driver	267
25.1 IWDG Firmware driver registers structures	267
25.1.1 IWDG_InitTypeDef	267
25.1.2 IWDG_HandleTypeDef	267
25.2 IWDG Firmware driver API description	267
25.2.1 IWDG Generic features	267
25.2.2 How to use this driver	268
25.2.3 Initialization and Start functions.....	268
25.2.4 IO operation functions	268
25.2.5 Detailed description of functions	268
25.3 IWDG Firmware driver defines	269
25.3.1 IWDG	269

26 HAL LCD Generic Driver	271
26.1 LCD Firmware driver registers structures.....	271
26.1.1 LCD_InitTypeDef.....	271
26.1.2 LCD_HandleTypeDef	272
26.2 LCD Firmware driver API description	272
26.2.1 How to use this driver	272
26.2.2 Initialization and Configuration functions.....	273
26.2.3 IO operation functions	273
26.2.4 Peripheral State functions	273
26.2.5 Detailed description of functions	274
26.3 LCD Firmware driver defines.....	276
26.3.1 LCD.....	276
27 HAL NOR Generic Driver.....	286
27.1 NOR Firmware driver registers structures.....	286
27.1.1 NOR_IDTypeDef	286
27.1.2 NOR_CFITTypeDef	286
27.1.3 NOR_HandleTypeDef.....	286
27.2 NOR Firmware driver API description	287
27.2.1 How to use this driver	287
27.2.2 NOR Initialization and de_initialization functions	287
27.2.3 NOR Input and Output functions	288
27.2.4 NOR Control functions.....	288
27.2.5 NOR State functions.....	288
27.2.6 Detailed description of functions	288
27.3 NOR Firmware driver defines.....	292
27.3.1 NOR.....	292
28 HAL OPAMP Generic Driver	294
28.1 OPAMP Firmware driver registers structures	294
28.1.1 OPAMP_InitTypeDef	294
28.1.2 OPAMP_HandleTypeDef.....	295
28.2 OPAMP Firmware driver API description	295
28.2.1 OPAMP Peripheral Features	295
28.2.2 How to use this driver	296
28.2.3 Initialization and de-initialization functions	297
28.2.4 IO operation functions	297
28.2.5 Peripheral Control functions	297

28.2.6	Peripheral State functions	297
28.2.7	Detailed description of functions	298
28.3	OPAMP Firmware driver defines.....	300
28.3.1	OPAMP.....	300
29	HAL OPAMP Extension Driver.....	302
29.1	OPAMPEx Firmware driver API description	302
29.1.1	Peripheral Control functions	302
29.1.2	Extended IO operation functions	302
29.1.3	Detailed description of functions	302
29.2	OPAMPEx Firmware driver defines.....	303
29.2.1	OPAMPEx	303
30	HAL PCD Generic Driver	304
30.1	PCD Firmware driver registers structures	304
30.1.1	PCD_InitTypeDef.....	304
30.1.2	PCD_EPTTypeDef.....	304
30.1.3	PCD_HandleTypeDef	305
30.2	PCD Firmware driver API description.....	306
30.2.1	How to use this driver	306
30.2.2	Initialization and de-initialization functions	306
30.2.3	IO operation functions	306
30.2.4	Peripheral Control functions	307
30.2.5	Peripheral State functions	307
30.2.6	Detailed description of functions	307
30.3	PCD Firmware driver defines	314
30.3.1	PCD	314
31	HAL PCD Extension Driver	316
31.1	PCDEx Firmware driver API description	316
31.1.1	Peripheral Control functions	316
31.1.2	Detailed description of functions	316
32	HAL PWR Generic Driver	317
32.1	PWR Firmware driver registers structures	317
32.1.1	PWR_PVTypeDef	317
32.2	PWR Firmware driver API description.....	317
32.2.1	Initialization and de-initialization functions	317
32.2.2	Peripheral Control functions	317
32.2.3	Detailed description of functions	321

32.3	PWR Firmware driver defines	325
32.3.1	PWR	325
33	HAL PWR Extension Driver	331
33.1	PWREx Firmware driver API description.....	331
33.1.1	Peripheral extended features functions.....	331
33.1.2	Detailed description of functions	331
33.2	PWREx Firmware driver defines	332
33.2.1	PWREx	332
34	HAL RCC Generic Driver.....	333
34.1	RCC Firmware driver registers structures	333
34.1.1	RCC_PLLInitTypeDef	333
34.1.2	RCC_OscInitTypeDef	333
34.1.3	RCC_ClkInitTypeDef	334
34.2	RCC Firmware driver API description	334
34.2.1	RCC specific features.....	334
34.2.2	RCC Limitations.....	335
34.2.3	Initialization and de-initialization functions	335
34.2.4	Peripheral Control functions	336
34.2.5	Detailed description of functions	336
34.3	RCC Firmware driver defines	341
34.3.1	RCC	341
35	HAL RCC Extension Driver	366
35.1	RCCEEx Firmware driver registers structures	366
35.1.1	RCC_PерiphCLKInitTypeDef	366
35.2	RCCEEx Firmware driver API description	366
35.2.1	Extended Peripheral Control functions.....	366
35.2.2	Detailed description of functions	366
35.3	RCCEEx Firmware driver defines.....	368
35.3.1	RCCEEx	368
36	HAL RTC Generic Driver	377
36.1	RTC Firmware driver registers structures	377
36.1.1	RTC_InitTypeDef	377
36.1.2	RTC_DateTypeDef	377
36.1.3	RTC_HandleTypeDef	378
36.2	RTC Firmware driver API description.....	378
36.2.1	Backup Domain Operating Condition	378

36.2.2	Backup Domain Reset.....	378
36.2.3	Backup Domain Access.....	379
36.2.4	How to use this driver	379
36.2.5	RTC and low power modes	379
36.2.6	Initialization and de-initialization functions	379
36.2.7	RTC Time and Date functions	380
36.2.8	RTC Alarm functions	380
36.2.9	Peripheral State functions	381
36.2.10	Peripheral Control functions	381
36.2.11	Detailed description of functions	381
36.3	RTC Firmware driver defines	387
36.3.1	RTC	387
37	HAL RTC Extension Driver	397
37.1	RTCEEx Firmware driver registers structures	397
37.1.1	RTC_TamperTypeDef	397
37.1.2	RTC_TimeTypeDef.....	397
37.1.3	RTC_AlarmTypeDef	398
37.2	RTCEEx Firmware driver API description.....	399
37.2.1	How to use this driver	399
37.2.2	RTC TimeStamp and Tamper functions.....	399
37.2.3	RTC Wake-up functions	400
37.2.4	Extension Peripheral Control functions	400
37.2.5	Extended features functions	401
37.2.6	Detailed description of functions	401
37.3	RTCEEx Firmware driver defines	410
37.3.1	RTCEEx	410
38	HAL SD Generic Driver	431
38.1	SD Firmware driver registers structures	431
38.1.1	SD_HandleTypeDef.....	431
38.1.2	HAL_SD_CSDTypedef.....	432
38.1.3	HAL_SD_CIDTypedef	434
38.1.4	HAL_SD_CardStatusTypedef	434
38.1.5	HAL_SD_CardInfoTypedef	435
38.2	SD Firmware driver API description	436
38.2.1	How to use this driver	436
38.2.2	Initialization and de-initialization functions	437
38.2.3	IO operation functions	438

38.2.4	Peripheral Control functions	438
38.2.5	Peripheral State functions	438
38.2.6	Detailed description of functions	438
38.3	SD Firmware driver defines.....	444
38.3.1	SD.....	444
39	HAL SMARTCARD Generic Driver.....	456
39.1	SMARTCARD Firmware driver registers structures	456
39.1.1	SMARTCARD_InitTypeDef	456
39.1.2	SMARTCARD_HandleTypeDef.....	457
39.2	SMARTCARD Firmware driver API description.....	458
39.2.1	How to use this driver	458
39.2.2	Initialization and Configuration functions.....	459
39.2.3	IO operation functions	460
39.2.4	Peripheral State and Errors functions	461
39.2.5	Detailed description of functions	461
39.3	SMARTCARD Firmware driver defines	465
39.3.1	SMARTCARD.....	465
40	HAL SPI Generic Driver.....	476
40.1	SPI Firmware driver registers structures	476
40.1.1	SPI_InitTypeDef	476
40.1.2	SPI_HandleTypeDef.....	477
40.2	SPI Firmware driver API description	477
40.2.1	How to use this driver	477
40.2.2	Initialization and de-initialization functions	478
40.2.3	IO operation functions	479
40.2.4	Peripheral State and Errors functions	479
40.2.5	Detailed description of functions	480
40.3	SPI Firmware driver defines	485
40.3.1	SPI.....	485
41	HAL SPI Extension Driver	491
41.1	SPIEx Firmware driver defines.....	491
41.1.1	SPIEx.....	491
42	HAL SRAM Generic Driver	492
42.1	SRAM Firmware driver registers structures.....	492
42.1.1	SRAM_HandleTypeDef.....	492
42.2	SRAM Firmware driver API description.....	492

42.2.1	How to use this driver	492
42.2.2	SRAM Initialization and de_initialization functions	493
42.2.3	SRAM Input and Output functions	493
42.2.4	SRAM Control functions	493
42.2.5	SRAM State functions	494
42.2.6	Detailed description of functions	494
42.3	SRAM Firmware driver defines	498
42.3.1	SRAM	498
43	HAL TIM Generic Driver	499
43.1	TIM Firmware driver registers structures.....	499
43.1.1	TIM_Base_InitTypeDef.....	499
43.1.2	TIM_OC_InitTypeDef.....	499
43.1.3	TIM_OnePulse_InitTypeDef	500
43.1.4	TIM_IC_InitTypeDef	500
43.1.5	TIM_Encoder_InitTypeDef	501
43.1.6	TIM_ClockConfigTypeDef	501
43.1.7	TIM_ClearInputConfigTypeDef.....	502
43.1.8	TIM_SlaveConfigTypeDef	502
43.1.9	TIM_HandleTypeDef	502
43.2	TIM Firmware driver API description	503
43.2.1	TIMER Generic features.....	503
43.2.2	How to use this driver	503
43.2.3	Time Base functions	504
43.2.4	Time Output Compare functions	505
43.2.5	Time PWM functions	505
43.2.6	Time Input Capture functions	505
43.2.7	Time One Pulse functions	506
43.2.8	Time Encoder functions.....	506
43.2.9	IRQ handler management	507
43.2.10	Peripheral Control functions	507
43.2.11	TIM Callbacks functions	507
43.2.12	Peripheral State functions	508
43.2.13	Detailed description of functions	508
43.3	TIM Firmware driver defines.....	531
43.3.1	TIM.....	531
44	HAL TIM Extension Driver.....	547
44.1	TIMEx Firmware driver registers structures.....	547
44.1.1	TIM_MasterConfigTypeDef	547

44.2	TIMEx Firmware driver API description	547
44.2.1	TIMER Extended features	547
44.2.2	Peripheral Control functions	547
44.2.3	Detailed description of functions	547
44.3	TIMEx Firmware driver defines	549
44.3.1	TIMEX	549
45	HAL UART Generic Driver.....	551
45.1	UART Firmware driver registers structures	551
45.1.1	UART_InitTypeDef	551
45.1.2	UART_HandleTypeDef	551
45.2	UART Firmware driver API description	552
45.2.1	How to use this driver	552
45.2.2	Initialization and Configuration functions	554
45.2.3	IO operation functions	555
45.2.4	Peripheral Control functions	556
45.2.5	Peripheral State and Errors functions	556
45.2.6	Detailed description of functions	557
45.3	UART Firmware driver defines	564
45.3.1	UART	564
46	HAL USART Generic Driver	574
46.1	USART Firmware driver registers structures	574
46.1.1	USART_InitTypeDef	574
46.1.2	USART_HandleTypeDef	574
46.2	USART Firmware driver API description	575
46.2.1	How to use this driver	575
46.2.2	Initialization and Configuration functions	577
46.2.3	IO operation functions	578
46.2.4	Peripheral State and Errors functions	579
46.2.5	Detailed description of functions	579
46.3	USART Firmware driver defines	585
46.3.1	USART	585
47	HAL WWDG Generic Driver	593
47.1	WWDG Firmware driver registers structures	593
47.1.1	WWDG_InitTypeDef	593
47.1.2	WWDG_HandleTypeDef	593
47.2	WWDG Firmware driver API description	593

Contents	UM1816
47.2.1 WWDG specific features	593
47.2.2 How to use this driver	594
47.2.3 Initialization and Configuration functions	594
47.2.4 IO operation functions	595
47.2.5 Detailed description of functions	595
47.3 WWDG Firmware driver defines.....	596
47.3.1 WWDG.....	596
48 LL ADC Generic Driver.....	599
48.1 ADC Firmware driver registers structures	599
48.1.1 LL_ADC_CommonInitTypeDef.....	599
48.1.2 LL_ADC_InitTypeDef.....	599
48.1.3 LL_ADC_REG_InitTypeDef.....	600
48.1.4 LL_ADC_INJ_InitTypeDef	600
48.2 ADC Firmware driver API description.....	601
48.2.1 Detailed description of functions	601
48.3 ADC Firmware driver defines	662
48.3.1 ADC	662
49 LL BUS Generic Driver.....	703
49.1 BUS Firmware driver API description.....	703
49.1.1 Detailed description of functions	703
49.2 BUS Firmware driver defines	720
49.2.1 BUS	720
50 LL COMP Generic Driver.....	723
50.1 COMP Firmware driver registers structures	723
50.1.1 LL_COMP_InitTypeDef	723
50.2 COMP Firmware driver API description	723
50.2.1 Detailed description of functions	723
50.3 COMP Firmware driver defines	732
50.3.1 COMP	732
51 LL CORTEX Generic Driver.....	737
51.1 CORTEX Firmware driver API description	737
51.1.1 Detailed description of functions	737
51.2 CORTEX Firmware driver defines.....	744
51.2.1 CORTEX.....	744
52 LL CRC Generic Driver.....	747
52.1 CRC Firmware driver API description	747

52.1.1	Detailed description of functions	747
52.2	CRC Firmware driver defines	748
52.2.1	CRC	748
53	LL DAC Generic Driver.....	750
53.1	DAC Firmware driver registers structures	750
53.1.1	LL_DAC_InitTypeDef.....	750
53.2	DAC Firmware driver API description.....	750
53.2.1	Detailed description of functions	750
53.3	DAC Firmware driver defines	766
53.3.1	DAC	766
54	LL DMA Generic Driver	772
54.1	DMA Firmware driver registers structures.....	772
54.1.1	LL_DMA_InitTypeDef	772
54.2	DMA Firmware driver API description	773
54.2.1	Detailed description of functions	773
54.3	DMA Firmware driver defines.....	806
54.3.1	DMA.....	806
55	LL EXTI Generic Driver	810
55.1	EXTI Firmware driver registers structures.....	810
55.1.1	LL_EXTI_InitTypeDef	810
55.2	EXTI Firmware driver API description	810
55.2.1	Detailed description of functions	810
55.3	EXTI Firmware driver defines.....	825
55.3.1	EXTI.....	825
56	LL GPIO Generic Driver	827
56.1	GPIO Firmware driver registers structures.....	827
56.1.1	LL_GPIO_InitTypeDef	827
56.2	GPIO Firmware driver API description	827
56.2.1	Detailed description of functions	827
56.3	GPIO Firmware driver defines.....	842
56.3.1	GPIO.....	842
57	LL I2C Generic Driver.....	845
57.1	I2C Firmware driver registers structures	845
57.1.1	LL_I2C_InitTypeDef.....	845
57.2	I2C Firmware driver API description.....	845

Contents	UM1816
57.2.1 Detailed description of functions	845
57.3 I2C Firmware driver defines	876
57.3.1 I2C	876
58 LL I2S Generic Driver	881
58.1 I2S Firmware driver registers structures	881
58.1.1 LL_I2S_InitTypeDef	881
58.2 I2S Firmware driver API description	881
58.2.1 Detailed description of functions	881
58.3 I2S Firmware driver defines	895
58.3.1 I2S	895
59 LL IWDG Generic Driver	897
59.1 IWDG Firmware driver API description	897
59.1.1 Detailed description of functions	897
59.2 IWDG Firmware driver defines	900
59.2.1 IWDG	900
60 LL OPAMP Generic Driver	902
60.1 OPAMP Firmware driver registers structures	902
60.1.1 LL_OPAMP_InitTypeDef	902
60.2 OPAMP Firmware driver API description	902
60.2.1 Detailed description of functions	902
60.3 OPAMP Firmware driver defines	912
60.3.1 OPAMP	912
61 LL PWR Generic Driver	916
61.1 PWR Firmware driver API description	916
61.1.1 Detailed description of functions	916
61.2 PWR Firmware driver defines	926
61.2.1 PWR	926
62 LL RCC Generic Driver	928
62.1 RCC Firmware driver registers structures	928
62.1.1 LL_RCC_ClocksTypeDef	928
62.2 RCC Firmware driver API description	928
62.2.1 Detailed description of functions	928
62.3 RCC Firmware driver defines	953
62.3.1 RCC	953
63 LL RTC Generic Driver	960

63.1	RTC Firmware driver registers structures	960
63.1.1	LL_RTC_InitTypeDef.....	960
63.1.2	LL_RTC_TimeTypeDef.....	960
63.1.3	LL_RTC_DateTypeDef.....	961
63.1.4	LL_RTC_AlarmTypeDef	961
63.2	RTC Firmware driver API description.....	962
63.2.1	Detailed description of functions	962
63.3	RTC Firmware driver defines	1026
63.3.1	RTC	1026
64	LL SPI Generic Driver.....	1036
64.1	SPI Firmware driver registers structures	1036
64.1.1	LL_SPI_InitTypeDef	1036
64.2	SPI Firmware driver API description	1037
64.2.1	Detailed description of functions	1037
64.3	SPI Firmware driver defines	1055
64.3.1	SPI.....	1055
65	LL SYSTEM Generic Driver.....	1062
65.1	SYSTEM Firmware driver API description	1062
65.1.1	Detailed description of functions	1062
65.2	SYSTEM Firmware driver defines	1091
65.2.1	SYSTEM.....	1091
66	LL TIM Generic Driver	1097
66.1	TIM Firmware driver registers structures.....	1097
66.1.1	LL_TIM_InitTypeDef	1097
66.1.2	LL_TIM_OC_InitTypeDef.....	1097
66.1.3	LL_TIM_IC_InitTypeDef	1098
66.1.4	LL_TIM_ENCODER_InitTypeDef.....	1098
66.2	TIM Firmware driver API description	1099
66.2.1	Detailed description of functions	1099
66.3	TIM Firmware driver defines.....	1151
66.3.1	TIM.....	1151
67	LL USART Generic Driver	1161
67.1	USART Firmware driver registers structures.....	1161
67.1.1	LL_USART_InitTypeDef	1161
67.1.2	LL_USART_ClockInitTypeDef.....	1161
67.2	USART Firmware driver API description	1162

67.2.1	Detailed description of functions	1162
67.3	USART Firmware driver defines.....	1207
67.3.1	USART.....	1207
68	LL UTILS Generic Driver	1211
68.1	UTILS Firmware driver registers structures.....	1211
68.1.1	LL_UTILS_PLLInitTypeDef	1211
68.1.2	LL_UTILS_ClkInitTypeDef.....	1211
68.2	UTILS Firmware driver API description.....	1211
68.2.1	System Configuration functions.....	1211
68.2.2	Detailed description of functions	1212
68.3	UTILS Firmware driver defines.....	1215
68.3.1	UTILS.....	1215
69	LL WWDG Generic Driver	1216
69.1	WWDG Firmware driver API description	1216
69.1.1	Detailed description of functions	1216
69.2	WWDG Firmware driver defines.....	1220
69.2.1	WWDG.....	1220
70	Correspondence between API registers and API low-layer driver functions.....	1221
70.1	ADC	1221
70.2	BUS.....	1228
70.3	COMP	1236
70.4	CORTEX	1237
70.5	CRC	1238
70.6	DAC	1238
70.7	DMA	1240
70.8	EXTI.....	1243
70.9	GPIO	1244
70.10	I2C	1244
70.11	I2S.....	1248
70.12	IWDG	1249
70.13	OPAMP	1250
70.14	PWR.....	1251
70.15	RCC	1252
70.16	RTC.....	1255

70.17	SPI	1264
70.18	SYSTEM	1266
70.19	TIM.....	1279
70.20	USART.....	1286
70.21	WWDG.....	1291
71	FAQs.....	1293
72	Revision history	1297

List of tables

Table 1: Acronyms and definitions	24
Table 2: HAL driver files	26
Table 3: User-application files	27
Table 4: API classification	31
Table 5: List of devices supported by HAL drivers	32
Table 6: HAL API naming rules	35
Table 7: Macros handling interrupts and specific clock configurations	36
Table 8: Callback functions	37
Table 9: HAL generic APIs	38
Table 10: HAL extension APIs	39
Table 11: Define statements used for HAL configuration	43
Table 12: Description of GPIO_InitTypeDef structure	45
Table 13: Description of EXTI configuration macros	47
Table 14: MSP functions	52
Table 15: Timeout values	55
Table 16: LL driver files	59
Table 17: Common peripheral initialization functions	61
Table 18: Optional peripheral initialization functions	62
Table 19: Specific Interrupt, DMA request and status flags management	64
Table 20: Available function formats	64
Table 21: Peripheral clock activation/deactivation management	64
Table 22: Peripheral activation/deactivation management	65
Table 23: Peripheral configuration management	65
Table 24: Peripheral register management	65
Table 25: Correspondence between ADC registers and ADC low-layer driver functions	1221
Table 26: Correspondence between BUS registers and BUS low-layer driver functions	1228
Table 27: Correspondence between COMP registers and COMP low-layer driver functions	1236
Table 28: Correspondence between CORTEX registers and CORTEX low-layer driver functions	1237
Table 29: Correspondence between CRC registers and CRC low-layer driver functions	1238
Table 30: Correspondence between DAC registers and DAC low-layer driver functions	1238
Table 31: Correspondence between DMA registers and DMA low-layer driver functions	1240
Table 32: Correspondence between EXTI registers and EXTI low-layer driver functions	1243
Table 33: Correspondence between GPIO registers and GPIO low-layer driver functions	1244
Table 34: Correspondence between I2C registers and I2C low-layer driver functions	1244
Table 35: Correspondence between I2S registers and I2S low-layer driver functions	1248
Table 36: Correspondence between IWDG registers and IWDG low-layer driver functions	1249
Table 37: Correspondence between OPAMP registers and OPAMP low-layer driver functions	1250
Table 38: Correspondence between PWR registers and PWR low-layer driver functions	1251
Table 39: Correspondence between RCC registers and RCC low-layer driver functions	1252
Table 40: Correspondence between RTC registers and RTC low-layer driver functions	1255
Table 41: Correspondence between SPI registers and SPI low-layer driver functions	1264
Table 42: Correspondence between SYSTEM registers and SYSTEM low-layer driver functions	1266
Table 43: Correspondence between TIM registers and TIM low-layer driver functions	1279
Table 44: Correspondence between USART registers and USART low-layer driver functions	1286
Table 45: Correspondence between WWDG registers and WWDG low-layer driver functions	1291
Table 46: Document revision history	1297

List of figures

Figure 1: Example of project template	28
Figure 2: Adding device-specific functions	40
Figure 3: Adding family-specific functions	40
Figure 4: Adding new peripherals	41
Figure 5: Updating existing APIs	41
Figure 6: File inclusion model	42
Figure 7: HAL driver model	50
Figure 8: Low Layer driver folders	60
Figure 9: Low Layer driver CMSIS files	61

1 Acronyms and definitions

Table 1: Acronyms and definitions

Acronym	Definition
ADC	Analog-to-digital converter
ANSI	American National Standards Institute
API	Application Programming Interface
BSP	Board Support Package
COMP	Comparator
CMSIS	Cortex Microcontroller Software Interface Standard
CPU	Central Processing Unit
CRYP	Cryptographic processor unit
CRC	CRC calculation unit
DAC	Digital to analog converter
DMA	Direct Memory Access
EXTI	External interrupt/event controller
FLASH	Flash memory
GPIO	General purpose I/Os
HAL	Hardware abstraction layer
I2C	Inter-integrated circuit
I2S	Inter-integrated sound
IRDA	InfraRed Data Association
IWDG	Independent watchdog
LCD	Liquid Crystal Display Controller
MSP	MCU Specific Package
NVIC	Nested Vectored Interrupt Controller
PCD	USB Peripheral Controller Driver
PWR	Power controller
RCC	Reset and clock controller
RNG	Random Number Generator
RTC	Real-time clock
SD	Secure Digital
SRAM	SRAM external memory
SMARTCARD	Smartcard IC
SPI	Serial Peripheral interface
SysTick	System tick timer
TIM	Advanced-control, general-purpose or basic timer
TSC	Touch Sensing Controller

Acronym	Definition
UART	Universal asynchronous receiver/transmitter
USART	Universal synchronous receiver/transmitter
WWDG	Window watchdog
USB	Universal Serial Bus
PPP	STM32 peripheral or block

2 Overview of HAL drivers

The HAL drivers were designed to offer a rich set of APIs and to interact easily with the application upper layers.

Each driver consists of a set of functions covering the most common peripheral features. The development of each driver is driven by a common API which standardizes the driver structure, the functions and the parameter names.

The HAL drivers include a set of driver modules, each module being linked to a standalone peripheral. However, in some cases, the module is linked to a peripheral functional mode. As an example, several modules exist for the USART peripheral: USART driver module, USART driver module, SMARTCARD driver module and IRDA driver module.

The HAL main features are the following:

- Cross-family portable set of APIs covering the common peripheral features as well as extension APIs in case of specific peripheral features.
- Three API programming models: polling, interrupt and DMA.
- APIs are RTOS compliant:
 - Fully reentrant APIs
 - Systematic usage of timeouts in polling mode.
- Support of peripheral multi-instance allowing concurrent API calls for multiple instances of a given peripheral (USART1, USART2...)
- All HAL APIs implement user-callback functions mechanism:
 - Peripheral Init/DeInit HAL APIs can call user-callback functions to perform peripheral system level Initialization/De-Initialization (clock, GPIOs, interrupt, DMA)
 - Peripherals interrupt events
 - Error events.
- Object locking mechanism: safe hardware access to prevent multiple spurious accesses to shared resources.
- Timeout used for all blocking processes: the timeout can be a simple counter or a timebase.

2.1 HAL and user-application files

2.1.1 HAL driver files

A HAL drivers are composed of the following set of files:

Table 2: HAL driver files

File	Description
<i>stm32l1xx_hal_ppp.c</i>	Main peripheral/module driver file. It includes the APIs that are common to all STM32 devices. <i>Example: stm32l1xx_hal_adc.c, stm32l1xx_hal_irda.c, ...</i>
<i>stm32lx_hal_ppp.h</i>	Header file of the main driver C file It includes common data, handle and enumeration structures, define statements and macros, as well as the exported generic APIs. <i>Example: stm32lx_hal_adc.h, stm32lx_hal_irda.h, ...</i>

File	Description
<i>stm32lx_xx_hal_ppp_ex.c</i>	Extension file of a peripheral/module driver. It includes the specific APIs for a given part number or family, as well as the newly defined APIs that overwrite the default generic APIs if the internal process is implemented in different way. <i>Example: stm32lx_xx_hal_adc_ex.c, stm32lx_xx_hal_dma_ex.c, ...</i>
<i>stm32lx_xx_hal_ppp_ex.h</i>	Header file of the extension C file. It includes the specific data and enumeration structures, define statements and macros, as well as the exported device part number specific APIs <i>Example: stm32lx_xx_hal_adc_ex.h, stm32lx_xx_hal_dma_ex.h, ...</i>
<i>stm32lx_xx_hal.c</i>	This file is used for HAL initialization and contains DBGMCU, Remap and Time Delay based on systick APIs.
<i>stm32lx_xx_hal.h</i>	<i>stm32lx_xx_hal.c</i> header file
<i>stm32lx_xx_hal_msp_template.c</i>	Template file to be copied to the user application folder. It contains the MSP initialization and de-initialization (main routine and callbacks) of the peripheral used in the user application.
<i>stm32lx_xx_hal_conf_template.h</i>	Template file allowing to customize the drivers for a given application.
<i>stm32lx_xx_hal_def.h</i>	Common HAL resources such as common define statements, enumerations, structures and macros.

2.1.2 User-application files

The minimum files required to build an application using the HAL are listed in the table below:

Table 3: User-application files

File	Description
<i>system_stm32lx_xx.c</i>	This file contains SystemInit() which is called at startup just after reset and before branching to the main program. It does not configure the system clock at startup (contrary to the standard library). This is to be done using the HAL APIs in the user files. It allows relocating the vector table in internal SRAM.
<i>startup_stm32lx_xx.s</i>	Toolchain specific file that contains reset handler and exception vectors. For some toolchains, it allows adapting the stack/heap size to fit the application requirements.
<i>stm32lx_xx_flash.icf (optional)</i>	Linker file for EWARM toolchain allowing mainly to adapt the stack/heap size to fit the application requirements.
<i>stm32lx_xx_hal_msp.c</i>	This file contains the MSP initialization and de-initialization (main routine and callbacks) of the peripheral used in the user application.
<i>stm32lx_xx_hal_conf.h</i>	This file allows the user to customize the HAL drivers for a specific application. It is not mandatory to modify this configuration. The application can use the default configuration without any modification.

File	Description
<i>stm32f1xx_it.c/.h</i>	<p>This file contains the exceptions handler and peripherals interrupt service routine, and calls HAL_IncTick() at regular time intervals to increment a local variable (declared in <i>stm32f1xx_hal.c</i>) used as HAL timebase. By default, this function is called each 1ms in SysTick ISR..</p> <p>The PPP_IRQHandler() routine must call HAL_PPP_IRQHandler() if an interrupt based process is used within the application.</p>
<i>main.c/.h</i>	<p>This file contains the main program routine, mainly:</p> <ul style="list-style-type: none"> • the call to HAL_Init() • assert_failed() implementation • system clock configuration • peripheral HAL initialization and user application code.

The STM32Cube package comes with ready-to-use project templates, one for each supported board. Each project contains the files listed above and a preconfigured project for the supported toolchains.

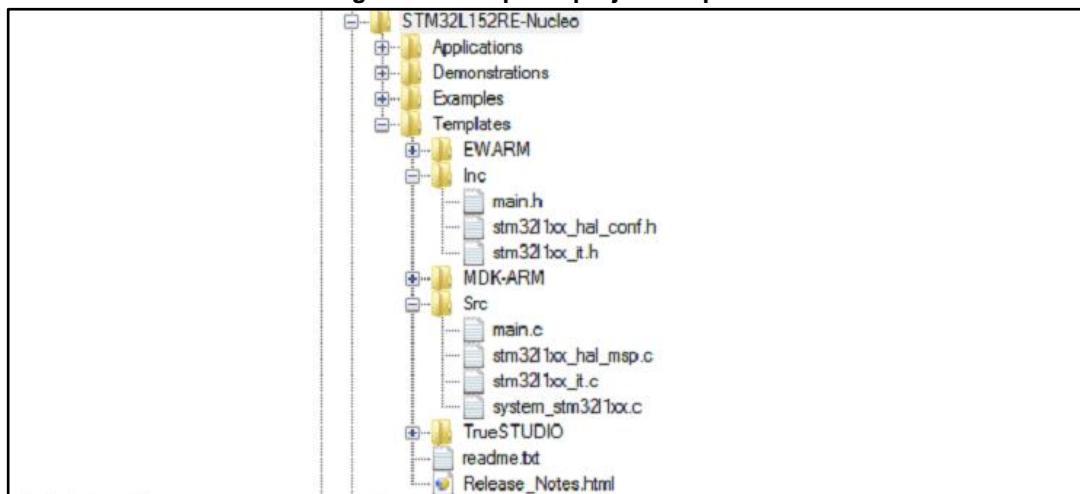
Each project template provides empty main loop function and can be used as a starting point to get familiar with project settings for STM32Cube. Its features are the following:

- It contains the sources of HAL, CMSIS and BSP drivers which are the minimal components to develop a code on a given board.
- It contains the include paths for all the firmware components.
- It defines the STM32 device supported, and allows configuring the CMSIS and HAL drivers accordingly.
- It provides ready to use user files preconfigured as defined below:
 - HAL is initialized
 - SysTick ISR implemented for HAL_Delay()
 - System clock configured with the maximum frequency of the device



If an existing project is copied to another location, then include paths must be updated.

Figure 1: Example of project template



2.2 HAL data structures

Each HAL driver can contain the following data structures:

- Peripheral handle structures
- Initialization and configuration structures
- Specific process structures.

2.2.1 Peripheral handle structures

The APIs have a modular generic multi-instance architecture that allows working with several IP instances simultaneously.

PPP_HandleTypeDef *handle is the main structure that is implemented in the HAL drivers. It handles the peripheral/module configuration and registers and embeds all the structures and variables needed to follow the peripheral device flow.

The peripheral handle is used for the following purposes:

- Multi instance support: each peripheral/module instance has its own handle. As a result instance resources are independent.
- Peripheral process intercommunication: the handle is used to manage shared data resources between the process routines.
Example: global pointers, DMA handles, state machine.
- Storage: this handle is used also to manage global variables within a given HAL driver.

An example of peripheral structure is shown below:

```
typedef struct
{
    USART_TypeDef *Instance; /* USART registers base address */
    USART_InitTypeDef Init; /* Usart communication parameters */
    uint8_t *pTxBuffPtr; /* Pointer to Usart Tx transfer Buffer */
    uint16_t TxXferSize; /* Usart Tx Transfer size */
    IO uint16_t TxXferCount; /* Usart Tx Transfer Counter */
    uint8_t *pRxBuffPtr; /* Pointer to Usart Rx transfer Buffer */
    uint16_t RxXferSize; /* Usart Rx Transfer size */
    IO uint16_t RxXferCount; /* Usart Rx Transfer Counter */
    DMA_HandleTypeDef *hdmatx; /* Usart Tx DMA Handle parameters */
    DMA_HandleTypeDef *hdmarx; /* Usart Rx DMA Handle parameters */
    HAL_LockTypeDef Lock; /* Locking object */
    IO HAL_USART_StateTypeDef State; /* Usart communication state */
    IO HAL_USART_ErrorTypeDef ErrorCode; /* USART Error code */
}USART_HandleTypeDef;
```



1) The multi-instance feature implies that all the APIs used in the application are re-entrant and avoid using global variables because subroutines can fail to be re-entrant if they rely on a global variable to remain unchanged but that variable is modified when the subroutine is recursively invoked. For this reason, the following rules are respected:

- Re-entrant code does not hold any static (or global) non-constant data: re-entrant functions can work with global data. For example, a re-entrant interrupt service routine can grab a piece of hardware status to work with (e.g. serial port read buffer) which is not only global, but volatile. Still, typical use of static variables and global data is not advised, in the sense that only atomic read-modify-write instructions should be used in these variables. It should not be possible for an interrupt or signal to occur during the execution of such an instruction.
- Reentrant code does not modify its own code.



2) When a peripheral can manage several processes simultaneously using the DMA (full duplex case), the DMA interface handle for each process is added in the PPP_HandleTypeDef.



3) For the shared and system peripherals, no handle or instance object is used. The peripherals concerned by this exception are the following:

- GPIO
- SYSTICK
- NVIC
- PWR
- RCC
- FLASH.

2.2.2 Initialization and configuration structure

These structures are defined in the generic driver header file when it is common to all part numbers. When they can change from one part number to another, the structures are defined in the extension header file for each part number.

```
typedef struct
{
    uint32_t BaudRate; /*!< This member configures the UART communication baudrate.*/
    uint32_t WordLength; /*!< Specifies the number of data bits transmitted or received
in a frame.*/
    uint32_t StopBits; /*!< Specifies the number of stop bits transmitted.*/
    uint32_t Parity; /*!< Specifies the parity mode. */
    uint32_t Mode; /*!< Specifies whether the Receive or Transmit mode is enabled or
disabled.*/
    uint32_t HwFlowCtl; /*!< Specifies whether the hardware flow control mode is enabled
or disabled.*/
    uint32_t OverSampling; /*!< Specifies whether the Over sampling 8 is enabled or
disabled,
to achieve higher speed (up to fPCLK/8).*/
}UART_HandleTypeDef;
```



The config structure is used to initialize the sub-modules or sub-instances. See below example:

```
HAL_ADC_ConfigChannel (ADC_HandleTypeDef* hadc, ADC_ChannelConfTypeDef*
sConfig)
```

2.2.3 Specific process structures

The specific process structures are used for specific process (common APIs). They are defined in the generic driver header file.

Example:

```
HAL_PPP_Process (PPP_HandleTypeDef* hadc, PPP_ProcessConfig* sConfig)
```

2.3 API classification

The HAL APIs are classified into three categories:

- **Generic APIs:** common generic APIs applying to all STM32 devices. These APIs are consequently present in the generic HAL driver files of all STM32 microcontrollers.

```

HAL_StatusTypeDef HAL_ADC_Init(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_DeInit(ADC_HandleTypeDef *hadc);
HAL_StatusTypeDef HAL_ADC_Start(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_Stop(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_Start_IT(ADC_HandleTypeDef* hadc);
HAL_StatusTypeDef HAL_ADC_Stop_IT(ADC_HandleTypeDef* hadc);
void HAL_ADC_IRQHandler(ADC_HandleTypeDef* hadc);

```

- **Extension APIs:** This set of API is divided into two sub-categories :
 - **Family specific APIs:** APIs applying to a given family. They are located in the extension HAL driver file (see example below related to the ADC).

```

HAL_StatusTypeDef HAL_ADCEx_Calibration_Start(ADC_HandleTypeDef* hadc, uint32_t t
SingleDiff);
uint32_t HAL_ADCEx_Calibration_GetValue(ADC_HandleTypeDef* hadc, uint32_t t
SingleDiff);

```

- **Device part number specific APIs:** These APIs are implemented in the extension file and delimited by specific define statements relative to a given part number.

```

#if defined(STM32L100xBA) || defined(STM32L151xBA) || defined(STM32L152xBA) || \
defined(STM32L100xC) || defined(STM32L151xC) || defined(STM32L152xC) || defined
(STM32L162xC) || \
defined(STM32L151xCA) || defined(STM32L151xD) || defined(STM32L152xCA) || defined
(STM32L152xD) || defined(STM32L162xCA) || defined(STM32L162xD) || \
defined(STM32L151xE) || defined(STM32L152xE) || defined(STM32L162xE)
void HAL_RCCEx_EnableLSECSS(void);
void HAL_RCCEx_DisableLSECSS(void);
#endif /* STM32L100xBA || .... STM32L162xE*/
The data structure related to the specific APIs is delimited by the device part
number define statement. It is located in the corresponding extension header C file.

```

The following table summarizes the location of the different categories of HAL APIs in the driver files.

Table 4: API classification

	Generic file	Extension file
Common APIs	X	X (1)
Family specific APIs		X
Device specific APIs		X

Notes:

(1) In some cases, the implementation for a specific device part number may change . In this case the generic API is declared as weak function in the extension file. The API is implemented again to overwrite the default function



Family specific APIs are only related to a given family. This means that if a specific API is implemented in another family, and the arguments of this latter family are different, additional structures and arguments might need to be added.



The IRQ handlers are used for common and family specific processes.

2.4 Devices supported by HAL drivers

Table 5: List of devices supported by HAL drivers

Files	VALUE LINE			ACCESS LINE					LCD LINE w/o AES								LCD Line w/ AES				
	STM32L100xB	STM32L100xBA	STM32L100xC	STM32L151xB	STM32L151xBA	STM32L151xC	STM32L151xCA	STM32L151xD	STM32L151xDX	STM32L151xE	STM32L152xB	STM32L152xBA	STM32L152xC	STM32L152xCA	STM32L152xD	STM32L152xDX	STM32L152xE	STM32L162xC	STM32L162xCA	STM32L162xD	STM32L162xDX
stm32l1xx_hal.c	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_adc.c stm32l1xx_hal_adc.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_adc_ex.c stm32l1xx_hal_adc_ex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_comp.c stm32l1xx_hal_comp.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_cortex.c stm32l1xx_hal_cortex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_crc.c stm32l1xx_hal_crc.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_cryp.c stm32l1xx_hal_cryp.h	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes
stm32l1xx_hal_dac.c stm32l1xx_hal_dac.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_dac_ex.c stm32l1xx_hal_dac_ex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_dma.c stm32l1xx_hal_dma.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_flash.c stm32l1xx_hal_flash.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_flash_ramfunc.c stm32l1xx_hal_flash_ramfunc.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_flash_ex.c stm32l1xx_hal_flash_ex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Files	VALUE LINE			ACCESS LINE						LCD LINE w/o AES						LCD Line w/ AES							
	STM32L100xB	STM32L100xBA	STM32L100xC	STM32L151xB	STM32L151xBA	STM32L151xC	STM32L151xCA	STM32L151xD	STM32L151xDX	STM32L151xE	STM32L152xB	STM32L152xBA	STM32L152xC	STM32L152xCA	STM32L152xD	STM32L152xDX	STM32L152xE	STM32L162xC	STM32L162xCA	STM32L162xD	STM32L162xDX	STM32L162xE	
stm32l1xx_hal_gpio.c stm32l1xx_hal_gpio.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_i2c.c stm32l1xx_hal_i2c.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_i2c_ex.c	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_i2s.c stm32l1xx_hal_i2s.h	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_irda.c stm32l1xx_hal_irda.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_iwdg.c stm32l1xx_hal_iwdg.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_lcd.c stm32l1xx_hal_lcd.h	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_msp_template.c	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
stm32l1xx_hal_nor.c stm32l1xx_hal_nor.h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No
stm32l1xx_hal_opamp.c stm32l1xx_hal_opamp.h	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_opamp_ex.c stm32l1xx_hal_opamp_ex.h	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_pcd.c stm32l1xx_hal_pcd.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_pcd_ex.c stm32l1xx_hal_pcd_ex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_pwr.c	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_pwr_ex.c stm32l1xx_hal_pwr_ex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_rcc.c stm32l1xx_hal_rcc.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Overview of HAL drivers

UM1816

Files	VALUE LINE			ACCESS LINE						LCD LINE w/o AES						LCD Line w/ AES							
	STM32L100xB	STM32L100xBA	STM32L100xC	STM32L151xB	STM32L151xBA	STM32L151xC	STM32L151xCA	STM32L151xD	STM32L151xDX	STM32L151xE	STM32L152xB	STM32L152xBA	STM32L152xC	STM32L152xCA	STM32L152xD	STM32L152xDX	STM32L152xE	STM32L162xC	STM32L162xCA	STM32L162xD	STM32L162xDX	STM32L162xE	
stm32l1xx_hal_rcc_ex.c stm32l1xx_hal_rcc_ex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal RTC.c stm32l1xx_hal_RTC.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_RTC_ex.c stm32l1xx_hal_RTC_ex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_sd.c stm32l1xx_hal_sd.h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No
stm32l1xx_hal_smartcard.c stm32l1xx_hal_smartcard.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_spi.c stm32l1xx_hal_spi.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_spi_ex.c	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_sram.c stm32l1xx_hal_sram.h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No
stm32l1xx_hal_tim.c stm32l1xx_hal_tim.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_tim_ex.cstm32l1xx_hal_tim_ex.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_uart.c stm32l1xx_hal_uart.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_usart.c stm32l1xx_hal_usart.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_wwdg.c stm32l1xx_hal_wwdg.h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_ll_fsmc.c stm32l1xx_ll_fsmc.h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No
stm32l1xx_ll_sdmmc.c stm32l1xx_ll_sdmmc.h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No

2.5 HAL driver rules

2.5.1 HAL API naming rules

The following naming rules are used in HAL drivers:

Table 6: HAL API naming rules

	Generic	Family specific	Device specific
File names	<i>stm32l1xx_hal_ppp (c/h)</i>	<i>stm32l1xx_hal_ppp_ex (c/h)</i>	<i>stm32l1xx_hal_ppp_ex (c/h)</i>
Module name	<i>HAL_PPP_MODULE</i>		
Function name	<i>HAL_PPP_Function</i> <i>HAL_PPP_FeatureFunction_MODE</i>	<i>HAL_PPPEx_Function</i> <i>HAL_PPPEx_FeatureFunction_MODE</i>	<i>HAL_PPPEx_Function</i> <i>HAL_PPPEx_FeatureFunction_MODE</i>
Handle name	<i>PPP_HandleTypeDef</i>	NA	NA
Init structure name	<i>PPP_InitTypeDef</i>	NA	<i>PPP_InitTypeDef</i>
Enum name	<i>HAL_PPP_StructnameTypeDef</i>	NA	NA

- The **PPP** prefix refers to the peripheral functional mode and not to the peripheral itself. For example, if the USART, PPP can be USART, IRDA, UART or SMARTCARD depending on the peripheral mode.
- The constants used in one file are defined within this file. A constant used in several files is defined in a header file. All constants are written in uppercase, except for peripheral driver function parameters.
- typedef variable names should be suffixed with _TypeDef.
- Registers are considered as constants. In most cases, their name is in uppercase and uses the same acronyms as in the STM32L1xx reference manuals.
- Peripheral registers are declared in the PPP_TypeDef structure (e.g. ADC_TypeDef) in stm32l1xxx.h header file. stm32l1xxx.h corresponds to stm32l100xb.h, stm32l100xba.h, stm32l100xc.h, stm32l151xb.h, stm32l151xba.h, stm32l151xc.h, stm32l151xca.h, stm32l151xd.h, stm32l151xe.h, stm32l151wdx.h, stm32l152xb.h, stm32l152xba.h, stm32l152xc.h, stm32l152xca.h, stm32l152xd.h, stm32l152xe.h, stm32l152wdx.h, stm32l162xc.h, stm32l162xca.h, stm32l162xd.h or stm32l162xe.h, stm32l162wdx.h.
- Peripheral function names are prefixed by HAL_, then the corresponding peripheral acronym in uppercase followed by an underscore. The first letter of each word is in uppercase (e.g. HAL_UART_Transmit()). Only one underscore is allowed in a function name to separate the peripheral acronym from the rest of the function name.
- The structure containing the PPP peripheral initialization parameters are named PPP_InitTypeDef (e.g. ADC_InitTypeDef).
- The structure containing the Specific configuration parameters for the PPP peripheral are named PPP_xxxxConfTypeDef (e.g. ADC_ChannelConfTypeDef).
- Peripheral handle structures are named PPP_HandleTypeDef (e.g DMA_HandleTypeDef)
- The functions used to initialize the PPP peripheral according to parameters specified in PPP_InitTypeDef are named HAL_PPP_Init (e.g. HAL_TIM_Init()).

- The functions used to reset the PPP peripheral registers to their default values are named PPP_Delnit, e.g. TIM_Delnit.
- The **MODE** suffix refers to the process mode, which can be polling, interrupt or DMA. As an example, when the DMA is used in addition to the native resources, the function should be called: *HAL_PPP_Function_DMA ()*.
- The **Feature** prefix should refer to the new feature. Example: *HAL_ADC_Start()* refers to the injection mode

2.5.2 HAL general naming rules

- For the shared and system peripherals, no handle or instance object is used. This rule applies to the following peripherals:
 - GPIO
 - SYSTICK
 - NVIC
 - RCC
 - FLASH.

Example: The *HAL_GPIO_Init()* requires only the GPIO address and its configuration parameters.

```
HAL_StatusTypeDef HAL_GPIO_Init (GPIO_TypeDef* GPIOx, GPIO_InitTypeDef *Init)
{
/*GPIO Initialization body */
}
```

- The macros that handle interrupts and specific clock configurations are defined in each peripheral/module driver. These macros are exported in the peripheral driver header files so that they can be used by the extension file. The list of these macros is defined below: This list is not exhaustive and other macros related to peripheral features can be added, so that they can be used in the user application.

Table 7: Macros handling interrupts and specific clock configurations

Macros	Description
<code>_HAL_PPP_ENABLE_IT(_HANDLE_, _INTERRUPT_)</code>	Enables a specific peripheral interrupt
<code>_HAL_PPP_DISABLE_IT(_HANDLE_, _INTERRUPT_)</code>	Disables a specific peripheral interrupt
<code>_HAL_PPP_GET_IT (_HANDLE_, _INTERRUPT_)</code>	Gets a specific peripheral interrupt status
<code>_HAL_PPP_CLEAR_IT (_HANDLE_, _INTERRUPT_)</code>	Clears a specific peripheral interrupt status
<code>_HAL_PPP_GET_FLAG (_HANDLE_, _FLAG_)</code>	Gets a specific peripheral flag status
<code>_HAL_PPP_CLEAR_FLAG (_HANDLE_, _FLAG_)</code>	Clears a specific peripheral flag status
<code>_HAL_PPP_ENABLE(_HANDLE_)</code>	Enables a peripheral
<code>_HAL_PPP_DISABLE(_HANDLE_)</code>	Disables a peripheral
<code>_HAL_PPP_XXXX (_HANDLE_, _PARAM_)</code>	Specific PPP HAL driver macro
<code>_HAL_PPP_GET_IT_SOURCE (_HANDLE_, _INTERRUPT_)</code>	Checks the source of specified interrupt

- NVIC and SYSTICK are two ARM Cortex core features. The APIs related to these features are located in the `stm32l1xx_hal_cortex.c` file.
- When a status bit or a flag is read from registers, it is composed of shifted values depending on the number of read values and of their size. In this case, the returned status width is 32 bits. Example : `STATUS = XX | (YY << 16)` or `STATUS = XX | (YY << 8) | (YY << 16) | (YY << 24)"`.
- The PPP handles are valid before using the `HAL_PPP_Init()` API. The init function performs a check before modifying the handle fields.

```
HAL_PPP_Init(PPP_HandleTypeDef)
if (hPPP == NULL)
{
    return HAL_ERROR;
}
```

- The macros defined below are used:
 - Conditional macro: `#define ABS(x) (((x) > 0) ? (x) : -(x))`
 - Pseudo-code macro (multiple instructions macro):

```
#define __HAL_LINKDMA(__HANDLE__, __PPP_DMA_FIELD__, __DMA_HANDLE__)
do{ \
    (__HANDLE__)->__PPP_DMA_FIELD__ = &(__DMA_HANDLE__); \
    (__DMA_HANDLE__).Parent = ( __HANDLE__ ); \
} while(0)
```

2.5.3 HAL interrupt handler and callback functions

Besides the APIs, HAL peripheral drivers include:

- `HAL_PPP_IRQHandler()` peripheral interrupt handler that should be called from `stm32l1xx_it.c`
- User callback functions.

The user callback functions are defined as empty functions with “weak” attribute. They have to be defined in the user code.

There are three types of user callbacks functions:

- Peripheral system level initialization/ de-Initialization callbacks: `HAL_PPP_MspInit()` and `HAL_PPP_MspDeInit()`
- Process complete callbacks : `HAL_PPP_ProcessCpltCallback`
- Error callback: `HAL_PPP_ErrorCallback`.

Table 8: Callback functions

Callback functions	Example
<code>HAL_PPP_MspInit() / _DeInit()</code>	Ex: <code>HAL_USART_MspInit()</code> Called from <code>HAL_PPP_Init()</code> API function to perform peripheral system level initialization (GPIOs, clock, DMA, interrupt)
<code>HAL_PPP_ProcessCpltCallback</code>	Ex: <code>HAL_USART_TxCpltCallback</code> Called by peripheral or DMA interrupt handler when the process completes
<code>HAL_PPP_ErrorCallback</code>	Ex: <code>HAL_USART_ErrorCallback</code> Called by peripheral or DMA interrupt handler when an error occurs

2.6 HAL generic APIs

The generic APIs provide common generic functions applying to all STM32 devices. They are composed of four APIs groups:

- **Initialization and de-initialization functions:** HAL_PPP_Init(), HAL_PPP_DeInit()
- **IO operation functions:** HAL_PPP_Read(), HAL_PPP_Write(), HAL_PPP_Transmit(), HAL_PPP_Receive()
- **Control functions:** HAL_PPP_Set(), HAL_PPP_Get().
- **State and Errors functions:** HAL_PPP_GetState(), HAL_PPP_GetError().

For some peripheral/module drivers, these groups are modified depending on the peripheral/module implementation.

Example: in the timer driver, the API grouping is based on timer features (PWM, OC, IC...).

The initialization and de-initialization functions allow initializing a peripheral and configuring the low-level resources, mainly clocks, GPIO, alternate functions (AF) and possibly DMA and interrupts. The *HAL_DeInit()* function restores the peripheral default state, frees the low-level resources and removes any direct dependency with the hardware.

The IO operation functions perform a row access to the peripheral payload data in write and read modes.

The control functions are used to change dynamically the peripheral configuration and set another operating mode.

The peripheral state and errors functions allow retrieving in runtime the peripheral and data flow states, and identifying the type of errors that occurred. The example below is based on the ADC peripheral. The list of generic APIs is not exhaustive. It is only given as an example.

Table 9: HAL generic APIs

Function group	Common API name	Description
<i>Initialization group</i>	<i>HAL_ADC_Init()</i>	This function initializes the peripheral and configures the low -level resources (clocks, GPIO, AF..)
	<i>HAL_ADC_DeInit()</i>	This function restores the peripheral default state, frees the low-level resources and removes any direct dependency with the hardware.
<i>IO operation group</i>	<i>HAL_ADC_Start()</i>	This function starts ADC conversions when the polling method is used
	<i>HAL_ADC_Stop()</i>	This function stops ADC conversions when the polling method is used
	<i>HAL_ADC_PollForConversion()</i>	This function allows waiting for the end of conversions when the polling method is used. In this case, a timeout value is specified by the user according to the application.
	<i>HAL_ADC_Start_IT()</i>	This function starts ADC conversions when the interrupt method is used
	<i>HAL_ADC_Stop_IT()</i>	This function stops ADC conversions when the interrupt method is used
	<i>HAL_ADC_IRQHandler()</i>	This function handles ADC interrupt requests

Function group	Common API name	Description
	<code>HAL_ADC_ConvCpltCallback()</code>	Callback function called in the IT subroutine to indicate the end of the current process or when a DMA transfer has completed
	<code>HAL_ADC_ErrorCallback()</code>	Callback function called in the IT subroutine if a peripheral error or a DMA transfer error occurred
<i>Control group</i>	<code>HAL_ADC_ConfigChannel()</code>	This function configures the selected ADC regular channel, the corresponding rank in the sequencer and the sample time
	<code>HAL_ADC_AnalogWDGConfig</code>	This function configures the analog watchdog for the selected ADC
<i>State and Errors group</i>	<code>HAL_ADC_GetState()</code>	This function allows getting in runtime the peripheral and the data flow states.
	<code>HAL_ADC_GetError()</code>	This function allows getting in runtime the error that occurred during IT routine

2.7 HAL extension APIs

2.7.1 HAL extension model overview

The extension APIs provide specific functions or overwrite modified APIs for a specific family (series) or specific part number within the same family.

The extension model consists of an additional file, `stm32l1xx_hal_ppp_ex.c`, that includes all the specific functions and define statements (`stm32l1xx_hal_ppp_ex.h`) for a given part number.

Below an example based on the ADC peripheral:

Table 10: HAL extension APIs

Function Group	Common API Name
<code>HAL_ADCEx_CalibrationStart()</code>	This function is used to start the automatic ADC calibration
<code>HAL_ADCEx_Calibration_GetValue()</code>	This function is used to get the ADC calibration factor
<code>HAL_ADCEx_Calibration_SetValue()</code>	This function is used to set the calibration factor to overwrite automatic conversion result

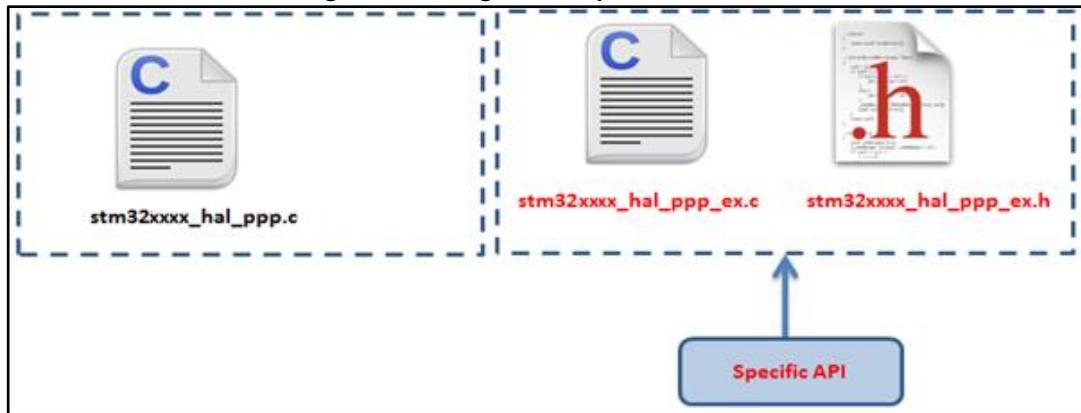
2.7.2 HAL extension model cases

The specific IP features can be handled by the HAL drivers in five different ways. They are described below.

Case 1: Adding a part number-specific function

When a new feature specific to a given device is required, the new APIs are added in the `stm32l1xx_hal_ppp_ex.c` extension file. They are named `HAL_PPPEX_Function()`.

Figure 2: Adding device-specific functions



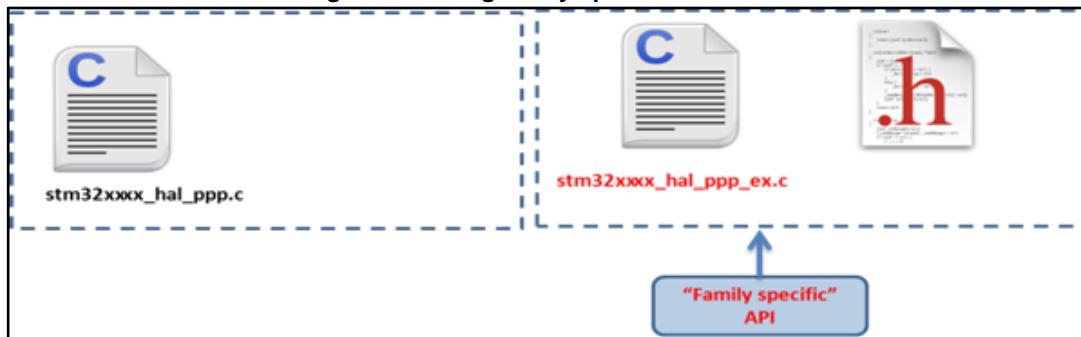
Example: `stm32l1xx_hal_rcc_ex.c/h`

```
#if defined(STM32L100xBA) || defined(STM32L151xBA) || defined(STM32L152xBA) || \
defined(STM32L100xC) || defined(STM32L151xC) || defined(STM32L152xC) || defined
STM32L162xC) || \
defined(STM32L151xCA) || defined(STM32L151xD) || defined(STM32L152xCA) || defined
(STM32L152xD) || defined(STM32L162xCA) || defined(STM32L162xD) || \
defined(STM32L151xE) || defined(STM32L152xE) || defined(STM32L162xE)
void HAL_RCCEx_EnableLSECSS(void);
void HAL_RCCEx_DisableLSECSS(void);
#endif /* STM32L100xBA || .... STM32L162xE*/
```

Case 2: Adding a family-specific function

In this case, the API is added in the extension driver C file and named `HAL_PPPEx_Function()`.

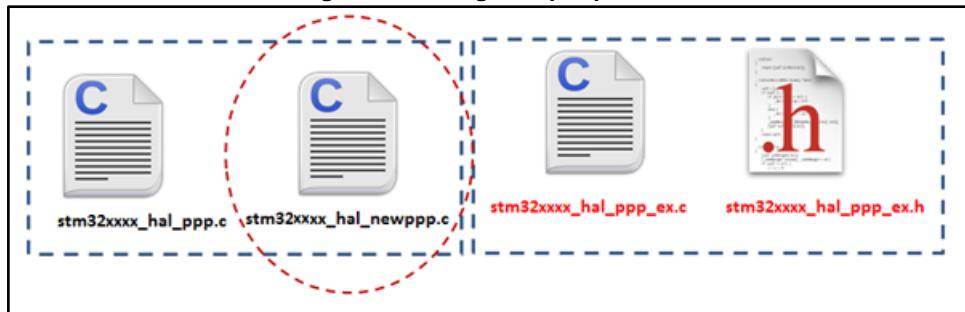
Figure 3: Adding family-specific functions



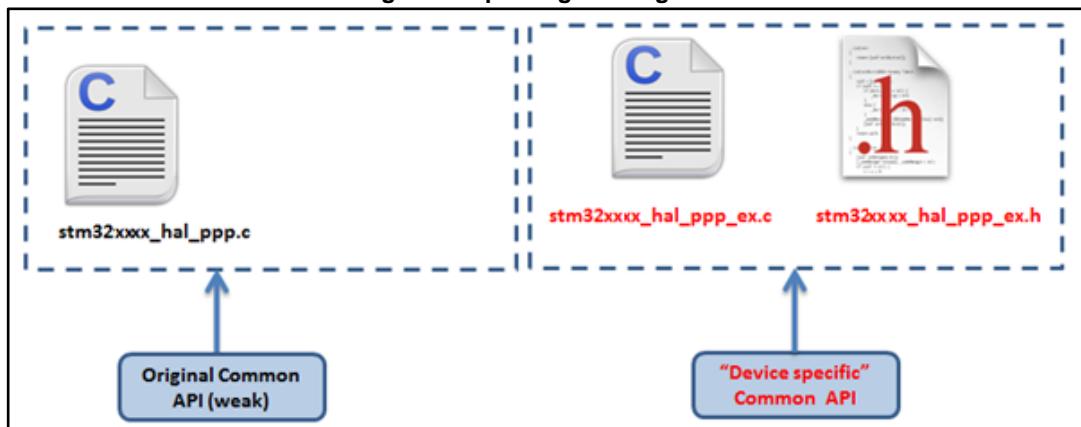
Case 3: Adding a new peripheral (specific to a device belonging to a given family)

When a peripheral which is available only in a specific device is required, the APIs corresponding to this new peripheral/module are added in `stm32l1xx_hal_newppp.c`. However the inclusion of this file is selected in the `stm32lx_xx_hal_conf.h` using the macro:

```
#define HAL_NEWPPP_MODULE_ENABLED
```

Figure 4: Adding new peripherals**Case 4: Updating existing common APIs**

In this case, the routines are defined with the same names in the `stm32l1xx_hal_ppp_ex.c` extension file, while the generic API is defined as *weak*, so that the compiler will overwrite the original routine by the new defined function.

Figure 5: Updating existing APIs**Case 5: Updating existing data structures**

The data structure for a specific device part number (e.g. `PPP_InitTypeDef`) can be composed of different fields. In this case, the data structure is defined in the extension header file and delimited by the specific part number define statement.

Example:

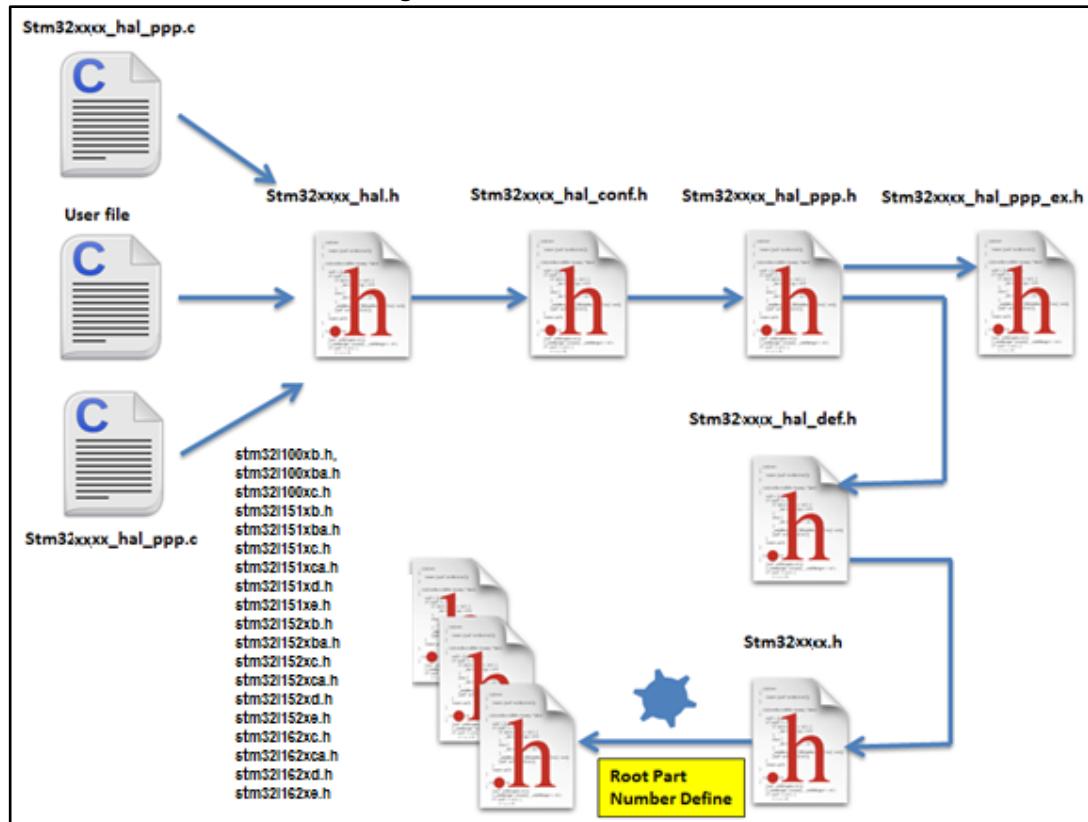
```
#if defined (STM32L100xB)
typedef struct
{
(...)

}PPP_InitTypeDef;
#endif /* STM32L100xB */
```

2.8**File inclusion model**

The header of the common HAL driver file (`stm32l1xx_hal.h`) includes the common configurations for the whole HAL library. It is the only header file that is included in the user sources and the HAL C sources files to be able to use the HAL resources.

Figure 6: File inclusion model



A PPP driver is a standalone module which is used in a project. The user must enable the corresponding USE_HAL_PPP_MODULE define statement in the configuration file.

```
/* ****
* @file stm3211xx_hal_conf.h
* @author MCD Application Team
* @version VX.Y.Z * @date dd-mm-yyyy
* @brief This file contains the modules to be used
**** */
(...)

#define USE_HAL_USART_MODULE
#define USE_HAL_IRDA_MODULE
#define USE_HAL_DMA_MODULE
#define USE_HAL_RCC_MODULE
(...)
```

2.9 HAL common resources

The common HAL resources, such as common define enumerations, structures and macros, are defined in `stm32l1xx_hal_def.h`. The main common define enumeration is `HAL_StatusTypeDef`.

- **HAL Status**

The HAL status is used by almost all HAL APIs, except for boolean functions and IRQ handler. It returns the status of the current API operations. It has four possible values as described below:

```
Typedef enum
{
    HAL_OK = 0x00,
    HAL_ERROR = 0x01,
    HAL_BUSY = 0x02,
```

```
HAL_TIMEOUT = 0x03
} HAL_StatusTypeDef;
```

- **HAL Locked**

The HAL lock is used by all HAL APIs to prevent accessing by accident shared resources.

```
typedef enum
{
    HAL_UNLOCKED = 0x00, /*!<Resources unlocked */
    HAL_LOCKED = 0x01 /*!< Resources locked */
} HAL_LockTypeDef; In addition to common resources, the stm32l1xx_hal_def.h file calls the stm32l1xx.h file in CMSIS library to get the data structures and the address mapping for all peripherals:
```

- Declarations of peripheral registers and bits definition.
- Macros to access peripheral registers hardware (Write register, Read register...etc.).

- **Common macros**

- Macro defining HAL_MAX_DELAY:

```
#define HAL_MAX_DELAY 0xFFFFFFFF
– Macro linking a PPP peripheral to a DMA structure pointer: __HAL_LINKDMA();
```

```
#define __HAL_LINKDMA( HANDLE , PPP DMA FIELD , DMA HANDLE ) \
do{ \
    ( __HANDLE__ )-> __PPP_DMA_FIELD__ = &( __DMA_HANDLE__ ); \
    ( __DMA_HANDLE__ ).Parent = ( __HANDLE__ ); \
} while(0)
```

2.10 HAL configuration

The configuration file, *stm32l1xx_hal_conf.h*, allows customizing the drivers for the user application. Modifying this configuration is not mandatory: the application can use the default configuration without any modification.

To configure these parameters, the user should enable, disable or modify some options by uncommenting, commenting or modifying the values of the related define statements as described in the table below:

Table 11: Define statements used for HAL configuration

Configuration item	Description	Default Value
HSE_VALUE	Defines the value of the external oscillator (HSE) expressed in Hz. The user must adjust this define statement when using a different crystal value.	8 000 000 (Hz)
HSE_STARTUP_TIMEOUT	Timeout for HSE start-up, expressed in ms	5000
HSI_VALUE	Defines the value of the internal oscillator (HSI) expressed in Hz.	16 000 000 (Hz)
MSI_VALUE	Defines the Internal Multiple Speed oscillator (MSI) value expressed in Hz.	2 097 000 (Hz)
LSE_VALUE	Defines the value of the external oscillator (HSE) expressed in Hz. The user must adjust this define statement when using a different crystal value.	32768 (Hz)
LSE_STARTUP_TIMEOUT	Timeout for LSE start-up, expressed in ms	5000
VDD_VALUE	VDD value	3300 (mV)
USERTOS	Enables the use of RTOS	FALSE (for future use)

Configuration item	Description	Default Value
PREFETCH_ENABLE	Enables prefetch feature	TRUE
BUFFER_CACHE_ENABLE	Enables buffer cache	FALSE



The `stm32l1xx_hal_conf_template.h` file is located in the HAL drivers *Inc* folder. It should be copied to the user folder, renamed and modified as described above.



By default, the values defined in the `stm32l1xx_hal_conf_template.h` file are the same as the ones used for the examples and demonstrations. All HAL include files are enabled so that they can be used in the user code without modifications.

2.11 HAL system peripheral handling

This chapter gives an overview of how the system peripherals are handled by the HAL drivers. The full API list is provided within each peripheral driver description section.

2.11.1 Clock

Two main functions can be used to configure the system clock:

- `HAL_RCC_OscConfig (RCC_OscInitTypeDef *RCC_OscInitStruct)`. This function configures/enables multiple clock sources (HSE, HSI, LSE, LSI, PLL).
- `HAL_RCC_ClockConfig (RCC_ClkInitTypeDef *RCC_ClkInitStruct, uint32_t FLatency)`. This function
 - selects the system clock source
 - configures AHB, APB1 and APB2 clock dividers
 - configures the number of Flash memory wait states
 - updates the SysTick configuration when HCLK clock changes.

Some peripheral clocks are not derived from the system clock (RTC, USB...). In this case, the clock configuration is performed by an extended API defined in `stm32l1xx_hal_rcc_ex.c`: `HAL_RCCEx_PeriphCLKConfig(RCC_PeriphCLKInitTypeDef *PeriphClkInit)`.

Additional RCC HAL driver functions are available:

- `HAL_RCC_DeInit()` Clock de-initialization function that returns clock configuration to reset state
- Get clock functions that allow retrieving various clock configurations (system clock, HCLK, PCLK1, PCLK2, ...)
- MCO and CSS configuration functions

A set of macros are defined in `stm32l1xx_hal_rcc.h` and `stm32l1xx_hal_rcc_ex.h`. They allow executing elementary operations on RCC block registers, such as peripherals clock gating/reset control:

- `__PPP_CLK_ENABLE/__PPP_CLK_DISABLE` to enable/disable the peripheral clock
- `__PPP_FORCE_RESET/__PPP_RELEASE_RESET` to force/release peripheral reset
- `__PPP_CLK_SLEEP_ENABLE/__PPP_CLK_SLEEP_DISABLE` to enable/disable the peripheral clock during low power (Sleep) mode.

2.11.2 GPIOs

GPIO HAL APIs are the following:

- HAL_GPIO_Init() / HAL_GPIO_DeInit()
- HAL_GPIO_ReadPin() / HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin () .

In addition to standard GPIO modes (input, output, analog), the pin mode can be configured as EXTI with interrupt or event generation.

When selecting EXTI mode with interrupt generation, the user must call HAL_GPIO_EXTI_IRQHandler() from stm32l1xx_it.c and implement HAL_GPIO_EXTI_Callback()

The table below describes the GPIO_InitTypeDef structure field.

Table 12: Description of GPIO_InitTypeDef structure

Structure field	Description
Pin	Specifies the GPIO pins to be configured. Possible values: GPIO_PIN_x or GPIO_PIN_All, where x[0..15]
Mode	Specifies the operating mode for the selected pins: GPIO mode or EXTI mode. Possible values are: <ul style="list-style-type: none"> • <u>GPIO mode</u> <ul style="list-style-type: none"> – GPIO_MODE_INPUT : Input floating – GPIO_MODE_OUTPUT_PP : Output push-pull – GPIO_MODE_OUTPUT_OD : Output open drain – GPIO_MODE_AF_PP : Alternate function push-pull – GPIO_MODE_AF_OD : Alternate function open drain – GPIO_MODE_ANALOG : Analog mode • <u>External Interrupt mode</u> <ul style="list-style-type: none"> – GPIO_MODE_IT_RISING : Rising edge trigger detection – GPIO_MODE_IT_FALLING : Falling edge trigger detection – GPIO_MODE_IT_RISING_FALLING : Rising/Falling edge trigger detection • <u>External Event mode</u> <ul style="list-style-type: none"> – GPIO_MODE_EVT_RISING : Rising edge trigger detection – GPIO_MODE_EVT_FALLING : Falling edge trigger detection – GPIO_MODE_EVT_RISING_FALLING: Rising/Falling edge trigger detection
Pull	Specifies the Pull-up or Pull-down activation for the selected pins. Possible values are: GPIO_NOPULL GPIO_PULLUP GPIO_PULLDOWN
Speed	Specifies the speed for the selected pins Possible values are: GPIO_SPEED_VERY_LOW GPIO_SPEED_LOW GPIO_SPEED_MEDIUM GPIO_SPEED_HIGH

Structure field	Description
Alternate	<p>Peripheral to be connected to the selected pins. Possible values: GPIO_AFx_PPP, where AFx: is the alternate function index PPP: is the peripheral instance Example: use GPIO_AF1_TIM2 to connect TIM2 IOs on AF1.</p> <p>These values are defined in the GPIO extended driver, since the AF mapping may change between product lines.</p>  <p>Refer to the “Alternate function mapping” table in the datasheets for the detailed description of the system and peripheral I/O alternate functions.</p>

Please find below typical GPIO configuration examples:

- Configuring GPIOs as output push-pull to drive external LEDs

```
GPIO_InitStruct.Pin = GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
```

- Configuring PA0 as external interrupt with falling edge sensitivity:

```
GPIO_InitStructure.Mode = GPIO_MODE_IT_FALLING;
GPIO_InitStructure.Pull = GPIO_NOPULL;
GPIO_InitStructure.Pin = GPIO_PIN_0;
HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);
```

- Configuring USART1 Tx (PA9, mapped on AF4) as alternate function:

```
GPIO_InitStruct.Pin = GPIO_PIN_9;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
GPIO_InitStruct.Alternate = GPIO_AF4_USART1;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

2.11.3 Cortex NVIC and SysTick timer

The Cortex HAL driver, stm32l1xx_hal_cortex.c, provides APIs to handle NVIC and SysTick. The supported APIs include:

- HAL_NVIC_SetPriority()
- HAL_NVIC_EnableIRQ() / HAL_NVIC_DisableIRQ()
- HAL_NVIC_SystemReset()
- HAL_SYSTICK_IRQHandler()
- HAL_NVIC_GetPendingIRQ() / HAL_NVIC_SetPendingIRQ() / HAL_NVIC_ClearPendingIRQ()
- HAL_SYSTICK_Config()
- HAL_SYSTICK_CLKSourceConfig()
- HAL_SYSTICK_Callback()

2.11.4 PWR

The PWR HAL driver handles power management. The features shared between all STM32 Series are listed below:

- PVD configuration, enabling/disabling and interrupt handling
 - HAL_PWR_PVDConfig()
 - HAL_PWR_EnablePVD() / HAL_PWR_DisablePVD()
 - HAL_PWR_PVD_IRQHandler()
 - HAL_PWR_PVDCALLBACK()
- Wakeup pin configuration
 - HAL_PWR_EnableWakeUpPin() / HAL_PWR_DisableWakeUpPin()
- Low-power mode entry
 - HAL_PWR_EnterSLEEPMode()
 - HAL_PWR_EnterSTOPMode()
 - HAL_PWR_EnterSTANDBYMode()

Depending on the STM32 Series, extension functions are available in `stm32l1xx_hal_pwr_ex`. Here are a few examples (the list is not exhaustive)

- Ultralow power mode control
 - HAL_PWREx_EnableUltraLowPower() / HAL_PWREx_DisableUltraLowPower()
 - HAL_PWREx_EnableLowPowerRunMode() /
HAL_PWREx_DisableLowPowerRunMode()

2.11.5 EXTI

The EXTI is not considered as a standalone peripheral but rather as a service used by other peripheral. As a result there are no EXTI APIs but each peripheral HAL driver implements the associated EXTI configuration and EXTI function are implemented as macros in its header file.

The first 16 EXTI lines connected to the GPIOs are managed within the GPIO driver. The `GPIO_InitTypeDef` structure allows configuring an I/O as external interrupt or external event.

The EXTI lines connected internally to the PVD, RTC, USB, and COMP are configured within the HAL drivers of these peripheral through the macros given in the table below. The EXTI internal connections depend on the targeted STM32 microcontroller (refer to the product datasheet for more details):

Table 13: Description of EXTI configuration macros

Macros	Description
<code>PPP_EXTI_LINE_FUNCTION</code>	Defines the EXTI line connected to the internal peripheral. Example: <code>#define PWR_EXTI_LINE_PVD ((uint32_t)0x00010000) /*!<External interrupt line 16 Connected to the PVD EXTI Line */</code>
<code>_HAL_PPP_EXTI_ENABLE_IT(_EXTI_LINE_)</code>	Enables a given EXTI line Example: <code>_HAL_PVD_EXTI_ENABLE_IT(PWR_EXTI_LINE_PVD)</code>
<code>_HAL_PPP_EXTI_DISABLE_IT(_EXTI_LINE_)</code>	Disables a given EXTI line. Example: <code>_HAL_PVD_EXTI_DISABLE_IT(PWR_EXTI_LINE_PVD)</code>

Macros	Description
<code>_HAL_PPP_EXTI_GET_FLAG(__EXTI_LINE__)</code>	Gets a given EXTI line interrupt flag pending bit status. Example: <code>_HAL_PVD_EXTI_GET_FLAG(PWR_EXTI_LINE_PVD)</code>
<code>_HAL_PPP_EXTI_CLEAR_FLAG(__EXTI_LINE__)</code>	Clears a given EXTI line interrupt flag pending bit. Example: <code>_HAL_PVD_EXTI_CLEAR_FLAG(PWR_EXTI_LINE_PVD)</code>
<code>_HAL_PPP_EXTI_GENERATE_SWIT(__EXTI_LINE__)</code>	Generates a software interrupt for a given EXTI line. Example: <code>_HAL_PVD_EXTI_GENERATE_SWIT(PWR_EXTI_LINE_PVD)</code>

If the EXTI interrupt mode is selected, the user application must call `HAL_PPP_FUNCTION_IRQHandler()` (for example `HAL_PWR_PVD_IRQHandler()`), from `stm32l1xx_it.c` file, and implement `HAL_PPP_FUNCTIONCallback()` callback function (for example `HAL_PWR_PVDCallback()`).

2.11.6 DMA

The DMA HAL driver allows enabling and configuring the peripheral to be connected to the DMA Channels (except for internal SRAM/FLASH memory which do not require any initialization). Refer to the product reference manual for details on the DMA request corresponding to each peripheral.

For a given channel, `HAL_DMA_Init()` API allows programming the required configuration through the following parameters:

- Transfer Direction
- Source and Destination data formats
- Circular, Normal or peripheral flow control mode
- Channels Priority level
- Source and Destination Increment mode

Two operating modes are available:

- Polling mode I/O operation
 - a. Use `HAL_DMA_Start()` to start DMA transfer when the source and destination addresses and the Length of data to be transferred have been configured.
 - b. Use `HAL_DMA_PollForTransfer()` to poll for the end of current transfer. In this case a fixed timeout can be configured depending on the user application.
- Interrupt mode I/O operation
 - a. Configure the DMA interrupt priority using `HAL_NVIC_SetPriority()`
 - b. Enable the DMA IRQ handler using `HAL_NVIC_EnableIRQ()`
 - c. Use `HAL_DMA_Start_IT()` to start DMA transfer when the source and destination addresses and the length of data to be transferred have been configured. In this case the DMA interrupt is configured.
 - d. Use `HAL_DMA_IRQHandler()` called under `DMA_IRQHandler()` Interrupt subroutine
 - e. When data transfer is complete, `HAL_DMA_IRQHandler()` function is executed and a user function can be called by customizing `XferCpltCallback` and `XferErrorCallback` function pointer (i.e. a member of DMA handle structure).

Additional functions and macros are available to ensure efficient DMA management:

- Use HAL_DMA_GetState() function to return the DMA state and HAL_DMA_GetError() in case of error detection.
- Use HAL_DMA_Abort() function to abort the current transfer

The most used DMA HAL driver macros are the following:

- __HAL_DMA_ENABLE: enables the specified DMA Channels.
- __HAL_DMA_DISABLE: disables the specified DMA Channels.
- __HAL_DMA_GET_FLAG: gets the DMA Channels pending flags.
- __HAL_DMA_CLEAR_FLAG: clears the DMA Channels pending flags.
- __HAL_DMA_ENABLE_IT: enables the specified DMA Channels interrupts.
- __HAL_DMA_DISABLE_IT: disables the specified DMA Channels interrupts.
- __HAL_DMA_GET_IT_SOURCE: checks whether the specified DMA stream interrupt has occurred or not.



When a peripheral is used in DMA mode, the DMA initialization should be done in the HAL_PPP_MspInit() callback. In addition, the user application should associate the DMA handle to the PPP handle (refer to section “HAL IO operation functions”).



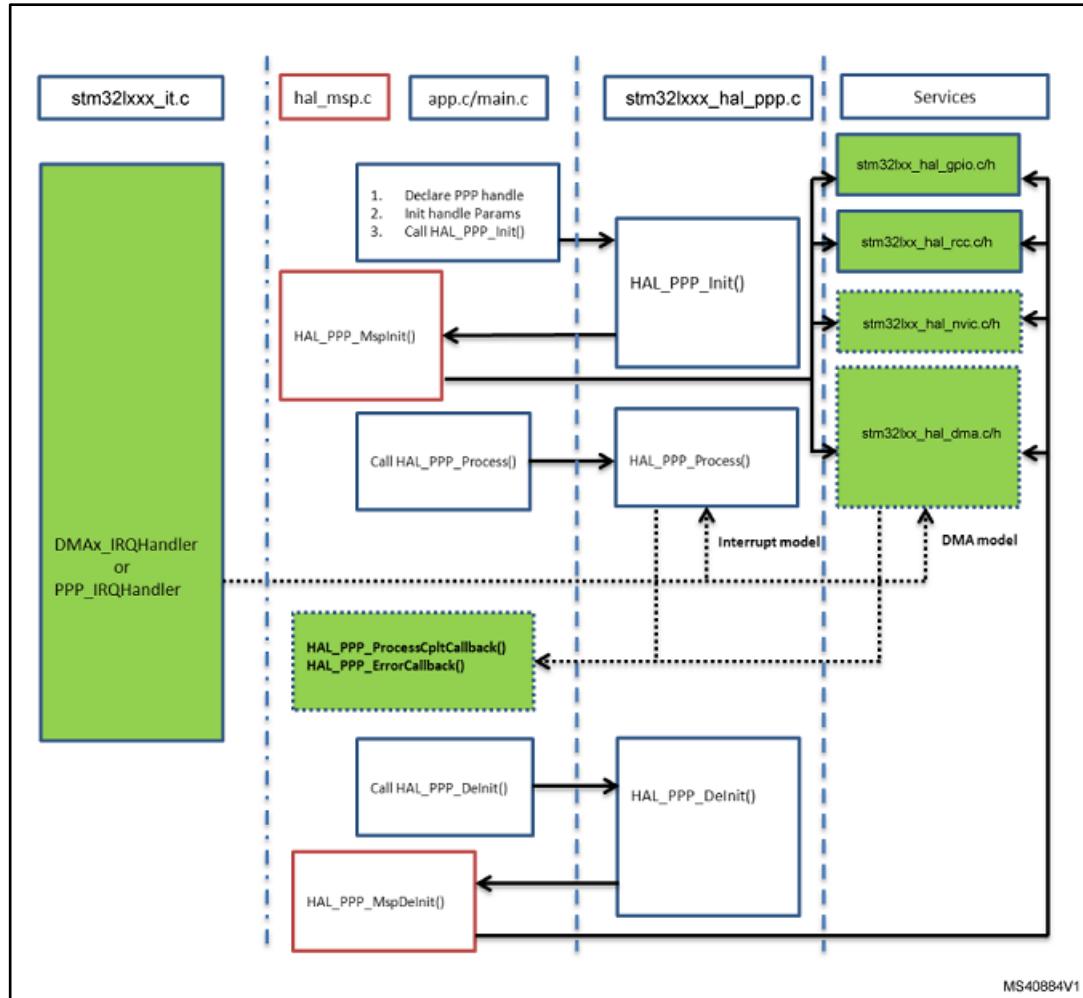
DMA channel callbacks need to be initialized by the user application only in case of memory-to-memory transfer. However when peripheral-to-memory transfers are used, these callbacks are automatically initialized by calling a process API function that uses the DMA.

2.12 How to use HAL drivers

2.12.1 HAL usage models

The following figure shows the typical use of the HAL driver and the interaction between the application user, the HAL driver and the interrupts.

Figure 7: HAL driver model



MS40884V1

Basically, the HAL driver APIs are called from user files and optionally from interrupt handlers file when the APIs based on the DMA or the PPP peripheral dedicated interrupts are used.

When DMA or PPP peripheral interrupts are used, the PPP process complete callbacks are called to inform the user about the process completion in real-time event mode (interrupts). Note that the same process completion callbacks are used for DMA in interrupt mode.

2.12.2 HAL initialization

2.12.2.1 HAL global initialization

In addition to the peripheral initialization and de-initialization functions, a set of APIs are provided to initialize the HAL core implemented in file `stm32l1xx_hal.c`.

- `HAL_Init()`: this function must be called at application startup to
 - initialize data/instruction cache and pre-fetch queue
 - set SysTick timer to generate an interrupt each 1ms (based on HSI clock) with the lowest priority
 - call `HAL_MspInit()` user callback function to perform system level initializations (Clock, GPIOs, DMA, interrupts). `HAL_MspInit()` is defined as “weak” empty function in the HAL drivers.
- `HAL_DeInit()`

- resets all peripherals
- calls function HAL_MspDeInit() which a is user callback function to do system level De-Initializations.
- HAL_GetTick(): this function gets current SysTick counter value (incremented in SysTick interrupt) used by peripherals drivers to handle timeouts.
- HAL_Delay(). this function implements a delay (expressed in milliseconds) using the SysTick timer.
Care must be taken when using HAL_Delay() since this function provides an accurate delay (expressed in milliseconds) based on a variable incremented in SysTick ISR. This means that if HAL_Delay() is called from a peripheral ISR, then the SysTick interrupt must have highest priority (numerically lower) than the peripheral interrupt, otherwise the caller ISR will be blocked.

2.12.2.2 System clock initialization

The clock configuration is done at the beginning of the user code. However the user can change the configuration of the clock in his own code. Please find below the typical Clock configuration sequence:

```
static void SystemClock_Config(void)
{
RCC_ClkInitTypeDef RCC_ClkInitStruct;
RCC_OscInitTypeDef RCC_OscInitStruct;
/* Enable MSI with range 5, PLL is OFF */
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
RCC_OscInitStruct.MSIRange = RCC_MSIRANGE_5;
RCC_OscInitStruct.MSICalibrationValue=0x00;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
HAL_RCC_OscConfig(&RCC_OscInitStruct);
/* Select MSI as system clock source and configure the HCLK, PCLK1 and PCLK2 clocks
dividers */
RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK |
RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
RCC_OscInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
RCC_OscInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_OscInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_OscInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0);
}
```

2.12.2.3 HAL MSP initialization process

The peripheral initialization is done through *HAL_PPP_Init()* while the hardware resources initialization used by a peripheral (PPP) is performed during this initialization by calling MSP callback function *HAL_PPP_MspInit()*.

The MspInit callback performs the low level initialization related to the different additional hardware resources: RCC, GPIO, NVIC and DMA.

All the HAL drivers with handles include two MSP callbacks for initialization and de-initialization:

```
/** 
* @brief Initializes the PPP MSP.
* @param hppp: PPP handle
* @retval None */
void __weak HAL_PPP_MspInit(PPP_HandleTypeDef *hppp) {
/* NOTE : This function Should not be modified, when the callback is needed,
the HAL PPP MspInit could be implemented in the user file */
}
/** 
* @brief DeInitializes PPP MSP.
* @param hppp: PPP handle
* @retval None */
void __weak HAL_PPP_MspDeInit(PPP_HandleTypeDef *hppp) {
```

```
/* NOTE : This function Should not be modified, when the callback is needed,
the HAL_PPP_MspDeInit could be implemented in the user file */
}
```

The MSP callbacks are declared empty as weak functions in each peripheral driver. The user can use them to set the low level initialization code or omit them and use his own initialization routine.

The HAL MSP callback is implemented inside the *stm32l1xx_hal_msp.c* file in the user folders. An *stm32l1xx_hal_msp.c* file template is located in the HAL folder and should be copied to the user folder. It can be generated automatically by STM32CubeMX tool and further modified. Note that all the routines are declared as weak functions and could be overwritten or removed to use user low level initialization code.

stm32l1xx_hal_msp.c file contains the following functions:

Table 14: MSP functions

Routine	Description
void HAL_MspInit()	Global MSP initialization routine
void HAL_MspDeInit()	Global MSP de-initialization routine
void HAL_PPP_MspInit()	PPP MSP initialization routine
void HAL_PPP_MspDeInit()	PPP MSP de-initialization routine

By default, if no peripheral needs to be de-initialized during the program execution, the whole MSP initialization is done in *Hal_MspInit()* and MSP De-Initialization in the *Hal_MspDeInit()*. In this case the *HAL_PPP_MspInit()* and *HAL_PPP_MspDeInit()* are not implemented.

When one or more peripherals needs to be de-initialized in run time and the low level resources of a given peripheral need to be released and used by another peripheral, *HAL_PPP_MspDeInit()* and *HAL_PPP_MspInit()* are implemented for the concerned peripheral and other peripherals initialization and de-Initialization are kept in the global *HAL_MspInit()* and the *HAL_MspDeInit()*.

If there is nothing to be initialized by the global *HAL_MspInit()* and *HAL_MspDeInit()*, the two routines can simply be omitted.

2.12.3 HAL IO operation process

The HAL functions with internal data processing like transmit, receive, write and read are generally provided with three data processing modes as follows:

- Polling mode
- Interrupt mode
- DMA mode

2.12.3.1 Polling mode

In Polling mode, the HAL functions return the process status when the data processing in blocking mode is complete. The operation is considered complete when the function returns the HAL_OK status, otherwise an error status is returned. The user can get more information through the *HAL_PPP_GetState()* function. The data processing is handled internally in a loop. A timeout (expressed in ms) is used to prevent process hanging.

The example below shows the typical Polling mode processing sequence :

```
HAL_StatusTypeDef HAL_PPP_Transmit ( PPP_HandleTypeDef * phandle, uint8_t pData,
int16_t tSize,uint32_t tTimeout)
```

```

{
if((pData == NULL ) || (Size == 0))
{
return HAL_ERROR;
}
(... ) while (data processing is running)
{
if( timeout reached )
{
return HAL_TIMEOUT;
}
}
(...)
return HELIAC; }
```

2.12.3.2 Interrupt mode

In Interrupt mode, the HAL function returns the process status after starting the data processing and enabling the appropriate interruption. The end of the operation is indicated by a callback declared as a weak function. It can be customized by the user to be informed in real-time about the process completion. The user can also get the process status through the *HAL_PPP_GetState()* function.

In interrupt mode, four functions are declared in the driver:

- *HAL_PPP_Process_IT()*: launch the process
- *HAL_PPP_IRQHandler()*: the global PPP peripheral interruption
- *__weak HAL_PPP_ProcessCpltCallback ()*: the callback relative to the process completion.
- *__weak HAL_PPP_ProcessErrorCallback()*: the callback relative to the process Error.

To use a process in interrupt mode, *HAL_PPP_Process_IT()* is called in the user file and *HAL_PPP_IRQHandler* in *stm32l1xx_it.c*.

The *HAL_PPP_ProcessCpltCallback()* function is declared as weak function in the driver. This means that the user can declare it again in the application. The function in the driver is not modified.

An example of use is illustrated below:

main.c file:

```

UART_HandleTypeDef UartHandle;
int main(void)
{
/* Set User Parameters */
UartHandle.Init.BaudRate = 9600;
UartHandle.Init.WordLength = UART_DATABITS_8;
UartHandle.Init.StopBits = UART_STOPBITS_1;
UartHandle.Init.Parity = UART_PARITY_NONE;
UartHandle.Init.HwFlowCtl = UART_HWCONTROL_NONE;
UartHandle.Init.Mode = UART_MODE_TX_RX;
UartHandle.Init.Instance = USART1;
HAL_UART_Init(&UartHandle);
HAL_UART_SendIT(&UartHandle, TxBuffer, sizeof(TxBuffer));
while (1);
}
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
}
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart)
{}
```

stm32l1xx_it.c file:



```

extern UART_HandleTypeDef UartHandle;
void USART1_IRQHandler(void)
{
  HAL_UART_IRQHandler(&UartHandle);
}

```

2.12.3.3 DMA mode

In DMA mode, the HAL function returns the process status after starting the data processing through the DMA and after enabling the appropriate DMA interruption. The end of the operation is indicated by a callback declared as a weak function and can be customized by the user to be informed in real-time about the process completion. The user can also get the process status through the *HAL_PPP_GetState()* function. For the DMA mode, three functions are declared in the driver:

- *HAL_PPP_Process_DMA()*: launch the process
- *HAL_PPP_DMA_IRQHandler()*: the DMA interruption used by the PPP peripheral
- *__weak HAL_PPP_ProcessCpltCallback()*: the callback relative to the process completion.
- *__weak HAL_PPP_ErrorCpltCallback()*: the callback relative to the process Error.

To use a process in DMA mode, *HAL_PPP_Process_DMA()* is called in the user file and the *HAL_PPP_DMA_IRQHandler()* is placed in the *stm32f1xx_it.c*. When DMA mode is used, the DMA initialization is done in the *HAL_PPP_MspInit()* callback. The user should also associate the DMA handle to the PPP handle. For this purpose, the handles of all the peripheral drivers that use the DMA must be declared as follows:

```

typedef struct
{
  PPP_HandleTypeDef *Instance; /* Register base address */
  PPP_InitTypeDef Init; /* PPP communication parameters */
  HAL_StatusTypeDef State; /* PPP communication state */
  ...
  DMA_HandleTypeDef *hdma; /* associated DMA handle */
} PPP_HandleTypeDef;

```

The initialization is done as follows (UART example):

```

int main(void)
{
/* Set User Parameters */
UartHandle.Init.BaudRate = 9600;
UartHandle.Init.WordLength = UART_DATABITS_8;
UartHandle.Init.StopBits = UART_STOPBITS_1;
UartHandle.Init.Parity = UART_PARITY_NONE;
UartHandle.Init.HwFlowCtl = UART_HWCONTROL_NONE;
UartHandle.Init.Mode = UART_MODE_TX_RX;
UartHandle.Init.Instance = USART1;
HAL_UART_Init(&UartHandle);
...
}
void HAL_USART_MspInit (UART_HandleTypeDef * huart)
{
  static DMA_HandleTypeDef hdma_tx;
  static DMA_HandleTypeDef hdma_rx;
  ...
  HAL_LINKDMA(UartHandle, DMA_HandleTypeDef_tx, hdma_tx);
  HAL_LINKDMA(UartHandle, DMA_HandleTypeDef_rx, hdma_rx);
  ...
}

```

The *HAL_PPP_ProcessCpltCallback()* function is declared as weak function in the driver that means, the user can declare it again in the application code. The function in the driver should not be modified.

An example of use is illustrated below:

main.c file:

```

UART_HandleTypeDef UartHandle;
int main(void)
{
/* Set User Parameters */
UartHandle.Init.BaudRate = 9600;
UartHandle.Init.WordLength = UART_DATABITS_8;
UartHandle.Init.StopBits = UART_STOPBITS_1;
UartHandle.Init.Parity = UART_PARITY_NONE;
UartHandle.Init.HwFlowCtl = UART_HWCONTROL_NONE;
UartHandle.Init.Mode = UART_MODE_TX_RX; UartHandle.Init.Instance = USART1;
HAL_UART_Init(&UartHandle);
HAL_UART_Send_DMA(&UartHandle, TxBuffer, sizeof(TxBuffer));
while (1);
}
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *phuart)
{
}
void HAL_UART_ErrorCallback(UART_HandleTypeDef *phuart)
{
}

```

stm32l1xx_it.c file:

```

extern UART_HandleTypeDef UartHandle;
void DMAx_IRQHandler(void)
{
HAL_DMA_IRQHandler(&UartHandle.DMA_Handle_tx);
}

```

HAL_USART_TxCpltCallback() and *HAL_USART_ErrorCallback()* should be linked in the *HAL_PPP_Process_DMA()* function to the DMA transfer complete callback and the DMA transfer Error callback by using the following statement:

```

HAL_PPP_Process_DMA (PPP_HandleTypeDef *hPPP, Params...)
{
(...)
hPPP->DMA_Handle->XferCpltCallback = HAL_UART_TxCpltCallback ;
hPPP->DMA_Handle->XferErrorCallback = HAL_UART_ErrorCallback ;
(... )
}

```

2.12.4 Timeout and error management

2.12.4.1 Timeout management

The timeout is often used for the APIs that operate in Polling mode. It defines the delay during which a blocking process should wait till an error is returned. An example is provided below:

```

HAL_StatusTypeDef HAL_DMA_PollForTransfer(DMA_HandleTypeDef *hdma, uint32_t
CompleteLevel, uint32_t Timeout)

```

The timeout possible value are the following:

Table 15: Timeout values

Timeout value	Description
0	No poll : Immediate process check and exit
1 ... (<i>HAL_MAX_DELAY</i> -1) ⁽¹⁾	Timeout in ms
<i>HAL_MAX_DELAY</i>	Infinite poll till process is successful

Notes:

⁽¹⁾*HAL_MAX_DELAY* is defined in the *stm32l1xx_hal_def.h* as 0xFFFFFFFF

However, in some cases, a fixed timeout is used for system peripherals or internal HAL driver processes. In these cases, the timeout has the same meaning and is used in the same way, except when it is defined locally in the drivers and cannot be modified or introduced as an argument in the user application.

Example of fixed timeout:

```
#define LOCAL_PROCESS_TIMEOUT 100
HAL_StatusTypeDef HAL_PPP_Process(PPP_HandleTypeDef)
{
(
...)
timeout = HAL_GetTick() + LOCAL_PROCESS_TIMEOUT;
(...)
while(ProcessOngoing)
{
(
...)
if(HAL_GetTick() >= timeout)
{
/* Process unlocked */
    HAL_UNLOCK(hppp);
hppp->State= HAL PPP STATE TIMEOUT;
return HAL PPP STATE TIMEOUT;
}
}
(...)
```

The following example shows how to use the timeout inside the polling functions:

```
HAL PPP StateTypeDef HAL PPP Poll (PPP_HandleTypeDef *hppp, uint32_t Timeout)
{
(
...)
timeout = HAL_GetTick() + Timeout;
(...)
while(ProcessOngoing)
{
(
...)
if(Timeout != HAL MAX DELAY)
{
if(HAL_GetTick() >= timeout)
{
/* Process unlocked */
    HAL_UNLOCK(hppp);
hppp->State= HAL PPP STATE TIMEOUT;
return hppp->State;
}
}
(...)
```

2.12.4.2 Error management

The HAL drivers implement a check on the following items:

- Valid parameters: for some process the used parameters should be valid and already defined, otherwise the system may crash or go into an undefined state. These critical parameters are checked before being used (see example below).

```
HAL_StatusTypeDef HAL_PPP_Process(PPP_HandleTypeDef* hppp, uint32_t *pdata, uint32
Size)
{
if ((pData == NULL) || (Size == 0))
{
return HAL_ERROR;
}
```

- Valid handle: the PPP peripheral handle is the most important argument since it keeps the PPP driver vital parameters. It is always checked in the beginning of the *HAL_PPP_Init()* function.

```
HAL_StatusTypeDef HAL_PPP_Init(PPP_HandleTypeDef* hppp)
{
if (hppp == NULL) //the handle should be already allocated
{
return HAL_ERROR;
}
```

- Timeout error: the following statement is used when a timeout error occurs: while (Process ongoing)

```
{
timeout = HAL_GetTick() + Timeout; while (data processing is running)
{
if(timeout) { return HAL_TIMEOUT;
}
}
```

When an error occurs during a peripheral process, *HAL_PPP_Process()* returns with a *HAL_ERROR* status. The HAL PPP driver implements the *HAL_PPP_GetError()* to allow retrieving the origin of the error.

```
HAL_PPP_ErrorTypeDef HAL_PPP_GetError (PPP_HandleTypeDef *hppp);
```

In all peripheral handles, a *HAL_PPP_ErrorTypeDef* is defined and used to store the last error code.

```
typedef struct
{
PPP_TypeDef * Instance; /* PPP registers base address */
PPP_InitTypeDef Init; /* PPP initialization parameters */
HAL_LockTypeDef Lock; /* PPP locking object */
IO_HandleTypeDef State; /* PPP state */
IO_HandleTypeDef ErrorCode; /* PPP Error code */
(...)

/* PPP specific parameters */
}
PPP_HandleTypeDef;
```

The error state and the peripheral global state are always updated before returning an error:

```
PPP->State = HAL_PPP_READY; /* Set the peripheral ready */
PP->ErrorCode = HAL_ERRORCODE ; /* Set the error code */
__HAL_UNLOCK(PPP) ; /* Unlock the PPP resources */
return HAL_ERROR; /*return with HAL error */
```

HAL_PPP_GetError() must be used in interrupt mode in the error callback:

```
void HAL_PPP_ProcessCpltCallback(PPP_HandleTypeDef *hspi)
{
ErrorCode = HAL_PPP_GetError (hppp); /* retreive error code */
}
```

2.12.4.3 Run-time checking

The HAL implements run-time failure detection by checking the input values of all HAL driver functions. The run-time checking is achieved by using an *assert_param* macro. This macro is used in all the HAL driver functions which have an input parameter. It allows verifying that the input value lies within the parameter allowed values.

To enable the run-time checking, use the *assert_param* macro, and leave the define **USE_FULL_ASSERT** uncommented in *stm32l1xx_hal_conf.h* file.

```

void HAL_UART_Init(UART_HandleTypeDef *huart)
{
    (...) /* Check the parameters */
    assert_param(IS_UART_INSTANCE(huart->Instance));
    assert_param(IS_UART_BAUDRATE(huart->Init.BaudRate));
    assert_param(IS_UART_WORD_LENGTH(huart->Init.WordLength));
    assert_param(IS_UART_STOPBITS(huart->Init.StopBits));
    assert_param(IS_UART_PARITY(huart->Init.Parity));
    assert_param(IS_UART_MODE(huart->Init.Mode));
    assert_param(IS_UART_HARDWARE_FLOW_CONTROL(huart->Init.HwFlowCtl));
    (...)

/** @defgroup UART_Word_Length *
@{
*/
#define UART_WORDLENGTH_8B ((uint32_t)0x00000000)
#define UART_WORDLENGTH_9B ((uint32_t)USART_CR1_M)
#define IS_UART_WORD_LENGTH(LENGTH) (((LENGTH) == UART_WORDLENGTH_8B) ||
\ ((LENGTH) == UART_WORDLENGTH_9B))

```

If the expression passed to the `assert_param` macro is false, the `assert_failed` function is called and returns the name of the source file and the source line number of the call that failed. If the expression is true, no value is returned.

The `assert_param` macro is implemented in `stm32l1xx_hal_conf.h`:

```

/* Exported macro -----*/
#ifndef USE_FULL_ASSERT
/**
 * @brief The assert param macro is used for function's parameters check.
 * @param expr: If expr is false, it calls assert failed function
 * which reports the name of the source file and the source
 * line number of the call that failed.
 * If expr is true, it returns no value.
 * @retval None */
#define assert_param(expr) ((expr)?(void)0:assert_failed((uint8_t *) FILE ,
LINE ))
/* Exported functions -----*/
void assert_failed(uint8_t* file, uint32_t line);
#endif /* USE_FULL_ASSERT */

```

The `assert_failed` function is implemented in the `main.c` file or in any other user C file:

```

#ifndef USE_FULL_ASSERT /**
 * @brief Reports the name of the source file and the source line number
 * where the assert param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None */
void assert_failed(uint8_t* file, uint32_t line)
{
/* User can add his own implementation to report the file name and line number,
ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* Infinite loop */
while (1)
{
}
}

```



Because of the overhead run-time checking introduces, it is recommended to use it during application code development and debugging, and to remove it from the final application to improve code size and speed.

3 Overview of Low Layer drivers

The Low Layer (LL) drivers are designed to offer a fast light-weight expert-oriented layer which is closer to the hardware than the HAL. Contrary to the HAL, LL APIs are not provided for peripherals where optimized access is not a key feature, or those requiring heavy software configuration and/or complex upper-level stack (such as FSMC, USB or SDMMC).

The LL drivers feature:

- A set of functions to initialize peripheral main features according to the parameters specified in data structures
- A set of functions used to fill initialization data structures with the reset values of each field
- Functions to perform peripheral de-initialization (peripheral registers restored to their default values)
- A set of inline functions for direct and atomic register access
- Full independence from HAL since LL drivers can be used either in standalone mode (without HAL drivers) or in mixed mode (with HAL drivers)
- Full coverage of the supported peripheral features.

The Low Layer drivers provide hardware services based on the available features of the STM32 peripherals. These services reflect exactly the hardware capabilities and provide one-shot operations that must be called following the programming model described in the microcontroller line reference manual. As a result, the LL services do not implement any processing and do not require any additional memory resources to save their states, counter or data pointers: all the operations are performed by changing the associated peripheral registers content.

3.1 Low Layer files

The Low Layer drivers are built around header/C files (one per each supported peripheral) plus five header files for some System and Cortex related features.

Table 16: LL driver files

File	Description
<i>stm32l1xx_ll_bus.h</i>	This is the h-source file for core bus control and peripheral clock activation and deactivation <i>Example: LL_AHB1_GRP1_EnableClock</i>
<i>stm32l1xx_ll_ppp.h/c</i>	<i>stm32l1xx_ll_ppp.c</i> provides peripheral initialization functions such as LL_PPP_Init(), LL_PPP_StructInit(), LL_PPP_DelInit(). All the other APIs are defined within <i>stm32l1xx_ll_ppp.h</i> file. The Low Layer PPP driver is a standalone module. To use it, the application must include it in the <i>stm32l1xx_ll_ppp.h</i> file.
<i>stm32l1xx_ll_cortex.h</i>	Cortex-M related register operation APIs including the Systick, Low power (LL_SYSTICK_xxxxx, LL_LPM_xxxxx "Low Power Mode" ...)
<i>stm32l1xx_ll_utils.h/c</i>	This file covers the generic APIs: <ul style="list-style-type: none"> • Read of device unique ID and electronic signature • Timebase and delay management • System clock configuration.
<i>stm32l1xx_ll_system.h</i>	System related operations (LL_SYS_CFG_xxx, LL_DBGMCU_xxx and LL_FLASH_xxx and LL_RI_xxx)

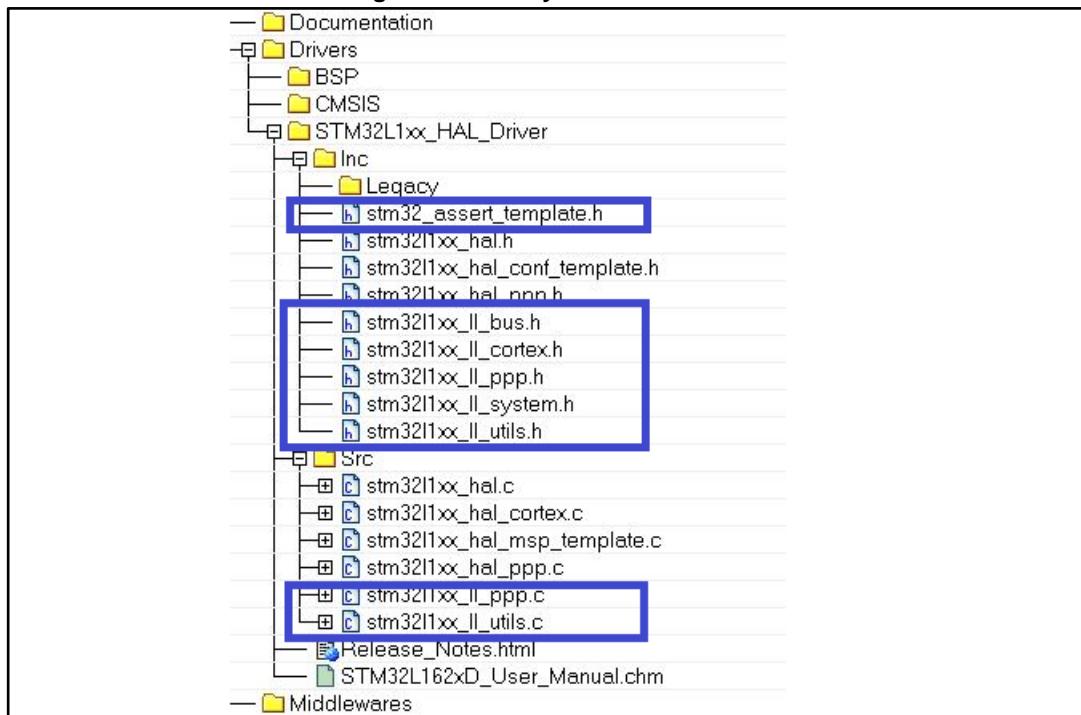
File	Description
stm32_assert_template.h	Template file allowing to define the assert_param macro, that is used when run-time checking is enabled. This file is required only when the LL drivers are used in standalone mode (without calling the HAL APIs). It should be copied to the application folder and renamed to stm32_assert.h.



There is no configuration file for the LL drivers.

The Low Layer files are located in the same HAL driver folder.

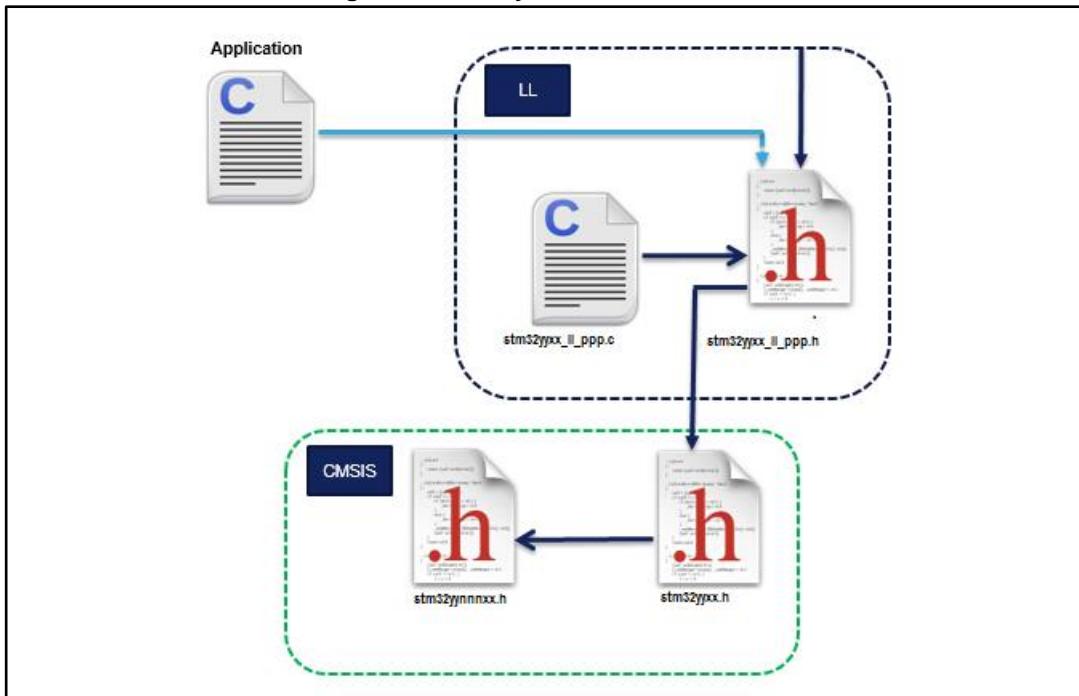
Figure 8: Low Layer driver folders



In general, Low Layer drivers include only the STM32 CMSIS device file.

```
#include "stm32yyxx.h"
```

Figure 9: Low Layer driver CMSIS files



Application files have to include only the used Low Layer drivers header files.

3.2 Overview of Low Layer APIs and naming rules

3.2.1 Peripheral initialization functions

The LL drivers offer three sets of initialization functions. They are defined in `stm32l1xx_ll_ppp.c` file:

- Functions to initialize peripheral main features according to the parameters specified in data structures
- A set of functions used to fill initialization data structures with the reset values of each field
- Function for peripheral de-initialization (peripheral registers restored to their default values)

The definition of these LL initialization functions and associated resources (structure, literals and prototypes) is conditioned by a compilation switch: `USE_FULL_LL_DRIVER`. To use these functions, this switch must be added in the toolchain compiler preprocessor or to any generic header file which is processed before the LL drivers.

The below table shows the list of the common functions provided for all the supported peripherals:

Table 17: Common peripheral initialization functions

Functions	Return Type	Parameters	Description
<code>LL_PPP_Init</code>	<code>ErrorStatus</code>	<ul style="list-style-type: none"> • <code>PPP_TypeDef* PPPx</code> • <code>LL_PPP_InitTypeDef* PPP_InitStruct</code> 	<p>Initializes the peripheral main features according to the parameters specified in <code>PPP_InitStruct</code>.</p> <p>Example: <code>LL_USART_Init(USART_TypeDef* USARTx, LL_USART_InitTypeDef* USART_InitStruct)</code></p>

Functions	Return Type	Parameters	Description
LL_PPP_StructInit	void	<ul style="list-style-type: none"> • <i>LL_PPP_InitTypeDef* PPP_InitStruct</i> 	Fills each PPP_InitStruct member with its default value. Example. <code>LL_USART_StructInit(LL_USART_InitTypeDef *USART_InitStruct)</code>
LL_PPP_Delinit	ErrorStatus	<ul style="list-style-type: none"> • <i>PPP_TypeDef* PPPx</i> 	De-initializes the peripheral registers, that is restore them to their default reset values. Example. <code>LL_USART_DeInit(USART_TypeDef *USARTx)</code>

Additional functions are available for some peripherals (refer to [Table 18: "Optional peripheral initialization functions"](#)):

Table 18: Optional peripheral initialization functions

Functions	Return Type	Parameters	Examples
LL_PPP{CATEGORY}_Init	Error Status	<ul style="list-style-type: none"> • <i>PPP_TypeDef* PPPx</i> • <i>LL_PPP{CATEGORY}_InitTypeDef* PPP{CATEGORY}_InitStruct</i> 	<p>Initializes peripheral features according to the parameters specified in PPP_InitStruct.</p> <p>Example:</p> <p><code>LL_ADC_INJ_Init(ADC_TypeDef *ADCx, LL_ADC_INJ_InitTypeDef *ADC_INJ_InitStruct)</code></p> <p><code>LL_RTC_TIME_Init(RTC_TypeDef *RTCx, uint32_t RTC_Format, LL_RTC_TimeTypeDef *RTC_TimeStruct)</code></p> <p><code>LL_RTC_DATE_Init(RTC_TypeDef *RTCx, uint32_t RTC_Format, LL_RTC_DateTypeDef *RTC_DateStruct)</code></p> <p><code>LL_TIM_IC_Init(TIM_TypeDef* TIMx, uint32_t Channel, LL_TIM_IC_InitTypeDef* TIM_IC_InitStruct)</code></p> <p><code>LL_TIM_ENCODER_Init(TIM_TypeDef* TIMx, LL_TIM_ENCODER_InitTypeDef* TIM_EncoderInitStruct)</code></p>
LL_PPP{CATEGORY}_StructInit	void	<i>LL_PPP{CATEGORY}_InitTypeDef* PPP{CATEGORY}_InitStruct</i>	<p>Fills each <i>PPP{CATEGORY}_InitStruct</i> member with its default value.</p> <p>Example:</p> <p><code>LL_ADC_INJ_StructInit(LL_ADC_INJ_Init TypeDef *ADC_INJ_InitStruct)</code></p>

Functions	Return Type	Parameters	Examples
LL_PPP_CommonInit	Error Status	<ul style="list-style-type: none"> <i>PPP_TypeDef* PPPx</i> <i>LL_PPP_CommonInitTypeDef* PPP_CommonInitStruct</i> 	<p>Initializes the common features shared between different instances of the same peripheral.</p> <p>Example: LL_ADC_CommonInit(ADC_Common_TypeDef *ADCxy_COMMON, LL_ADC_CommonInitTypeDef *ADC_CommonInitStruct)</p>
LL_PPP_CommonStructInit	void	<i>LL_PPP_CommonInitTypeDef* PPP_CommonInitStruct</i>	<p>Fills each <i>PPP_CommonInitStruct</i> member with its default value</p> <p>Example: LL_ADC_CommonStructInit(LL_ADC_CommonInitTypeDef *ADC_CommonInitStruct)</p>
LL_PPP_ClockInit	ErrorStatus	<ul style="list-style-type: none"> <i>PPP_TypeDef* PPPx</i> <i>LL_PPP_ClockInitTypeDef* PPP_ClockInitStruct</i> 	<p>Initializes the peripheral clock configuration in synchronous mode.</p> <p>Example: LL_USART_ClockInit(USART_TypeDef *USARTx, LL_USART_ClockInitTypeDef *USART_ClockInitStruct)</p>
LL_PPP_ClockStructInit	void	<i>LL_PPP_ClockInitTypeDef* PPP_ClockInitStruct</i>	<p>Fills each <i>PPP_ClockInitStruct</i> member with its default value</p> <p>Example: LL_USART_ClockStructInit(LL_USART_ClockInitTypeDef *USART_ClockInitStruct)</p>

3.2.1.1 Run-time checking

Like HAL drivers, LL initialization functions implement run-time failure detection by checking the input values of all LL driver functions. For more details please refer to [Section 2.12.4.3: "Run-time checking"](#).

When using the LL drivers in standalone mode (without calling HAL functions), the following actions are required to use run-time checking:

1. Copy `stm32_assert_template.h` to the application folder and rename it to `stm32_assert.h`. This file defines the `assert_param` macro which is used when run-time checking is enabled.
2. Include `stm32_assert.h` file within the application main header file.
3. Add the `USE_FULL_ASSERT` compilation switch in the toolchain compiler preprocessor or in any generic header file which is processed before the `stm32_assert.h` driver.



Run-time checking is not available for LL inline functions.

3.2.2 Peripheral register-level configuration functions

On top of the peripheral initialization functions, the LL drivers offer a set of inline functions for direct atomic register access. Their format is as follows:

```
_STATIC_INLINE return_type LL_PPP_Function (PPPx_TypeDef *PPPx, args)
```

The “Function” naming is defined depending to the action category:

- **Specific Interrupt, DMA request and status flags management:**
Set/Get/Clear/Enable/Disable flags on interrupt and status registers

Table 19: Specific Interrupt, DMA request and status flags management

Name	Examples
<i>LL_PPP_{_CATEGORY}_ActionItem_BITNAME</i> <i>LL_PPP{CATEGORY}_IsItem_BITNAME_Action</i>	<ul style="list-style-type: none"> • LL_RCC_IsActiveFlag_LSIRDY • LL_RCC_IsActiveFlag_FWRST() • LL_ADC_ClearFlag_EOC(ADC1) • LL_DMA_ClearFlag_TCx(DMA_TypeDef* DMAx)

Table 20: Available function formats

Item	Action	Format
Flag	Get	<i>LL_PPP_IsActiveFlag_BITNAME</i>
	Clear	<i>LL_PPP_ClearFlag_BITNAME</i>
Interrupts	Enable	<i>LL_PPP_EnableIT_BITNAME</i>
	Disable	<i>LL_PPP_DisableIT_BITNAME</i>
	Get	<i>LL_PPP_IsEnabledIT_BITNAME</i>
DMA	Enable	<i>LL_PPP_EnableDMAReq_BITNAME</i>
	Disable	<i>LL_PPP_DisableDMAReq_BITNAME</i>
	Get	<i>LL_PPP_IsEnabledDMAReq_BITNAME</i>



BITNAME refers to the peripheral register bit name as described in the product line reference manual.

- **Peripheral clock activation/deactivation management:** Enable/Disable/Reset a peripheral clock

Table 21: Peripheral clock activation/deactivation management

Name	Examples
<i>LL_bus_GRPx_ActionClock{Mode}</i>	<ul style="list-style-type: none"> • <i>LL_AHB1_GRP1_EnableClock</i> (<i>LL_AHB1_GRP1_PERIPH_GPIOA LL_AHB1_GRP1_PERIPH_GPIOB</i>) • <i>LL_APB1_GRP1_EnableClockSleep</i> (<i>LL_APB1_GRP1_PERIPH_DAC1</i>)



'x' corresponds to the group index and refers to the index of the modified register on a given bus.



'bus' refers to the bus name (eg APB1).

- **Peripheral activation/deactivation management:** Enable/disable a peripheral or activate/deactivate specific peripheral features

Table 22: Peripheral activation/deactivation management

Name	Examples
<code>LL_PPP_{CATEGORY}_Action{Item}</code> <code>LL_PPP_{CATEGORY}_IsItemAction</code>	<ul style="list-style-type: none"> • <code>LL_ADC_Enable ()</code> • <code>LL_ADC_StartCalibration();</code> • <code>LL_ADC_IsCalibrationOnGoing;</code> • <code>LL_RCC_HSI_Enable ()</code> • <code>LL_RCC_HSI_IsReady()</code>

- **Peripheral configuration management:** Set/get a peripheral configuration settings

Table 23: Peripheral configuration management

Name	Examples
<code>LL_PPP_{CATEGORY}_Set{ or Get}ConfigItem</code>	<code>LL_USART_SetBaudRate (USART2, 16000000,</code> <code>LL_USART_OVERSAMPLING_16, 9600)</code>

- **Peripheral register management:** Write/read the content of a register/retrun DMA relative register address

Table 24: Peripheral register management

Name
<code>LL_PPP_WriteReg(__INSTANCE__, __REG__, __VALUE__)</code>
<code>LL_PPP_ReadReg(__INSTANCE__, __REG__)</code>
<code>LL_PPP_DMA_GetRegAddr (PPP_TypeDef *PPPx,{Sub Instance if any ex: Channel}, {uint32_t Proprietary})</code>



The Proprietary is a variable used to identify the DMA transfer direction or the data register type.

4 Cohabiting of HAL and LL

The Low Layer is designed to be used in standalone mode or combined with the HAL. It cannot be automatically used with the HAL for the same peripheral instance. If you use the LL APIs for a specific instance, you can still use the HAL APIs for other instances. Be careful that the Low Layer might overwrite some registers which content is mirrored in the HAL handles.

4.1 Low Layer driver used in standalone mode

The Low Layer APIs can be used without calling the HAL driver services. This is done by simply including `stm32l1xx_ll_ppp.h` in the application files. The LL APIs for a given peripheral are called by executing the same sequence as the one recommended by the programming model in the corresponding product line reference manual. In this case the HAL drivers associated to the used peripheral can be removed from the workspace. However the STM32CubeL1 framework should be used in the same way as in the HAL drivers case which means that System file, startup file and CMSIS should always be used.



When the BSP drivers are included, the used HAL drivers associated with the BSP functions drivers should be included in the workspace, even if they are not used by the application layer.

4.2 Mixed use of Low Layer APIs and HAL drivers

In this case the Low Layer APIs are used in conjunction with the HAL drivers to achieve direct and register level based operations.

Mixed use is allowed, however some consideration should be taken into account:

- It is recommended to avoid using simultaneously the HAL APIs and the combination of Low Layer APIs for a given peripheral instance. If this is the case, one or more private fields in the HAL PPP handle structure should be updated accordingly.
- For operations and processes that do not alter the handle fields including the initialization structure, the HAL driver APIs and the Low Layer services can be used together for the same peripheral instance.
- The Low Layer drivers can be used without any restriction with all the HAL drivers that are not based on handle objects (RCC, common HAL, flash and GPIO).

Several examples showing how to use HAL and LL in the same application are provided within STM32L1 firmware package (refer to Examples_MIX projects).



1. When the HAL Init/DeInit APIs are not used and are replaced by the Low Layer macros, the `InitMsp()` functions are not called and the MSP initialization should be done in the user application.
2. When process APIs are not used and the corresponding function is performed through the Low Layer APIs, the callbacks are not called and post processing or error management should be done by the user application.
3. When the LL APIs are used for process operations, the IRQ handler HAL APIs cannot be called and the IRQ should be implemented by the user application. Each LL driver implements the macros needed to read and clear the associated interrupt flags.

5 HAL System Driver

5.1 HAL Firmware driver API description

5.1.1 How to use this driver

The common HAL driver contains a set of generic and common APIs that can be used by the PPP peripheral drivers and the user to start using the HAL.

The HAL contains two APIs' categories:

- Common HAL APIs
- Services HAL APIs

5.1.2 Initialization and de-initialization functions

This section provides functions allowing to:

- Initializes the Flash interface, the NVIC allocation and initial clock configuration. It initializes the source of time base also when timeout is needed and the backup domain when enabled.
- de-Initializes common part of the HAL.
- Configure The time base source to have 1ms time base with a dedicated Tick interrupt priority.
 - Systick timer is used by default as source of time base, but user can eventually implement his proper time base source (a general purpose timer for example or other time source), keeping in mind that Time base duration should be kept 1ms since PPP_TIMEOUT_VALUES are defined and handled in milliseconds basis.
 - Time base configuration function (HAL_InitTick ()) is called automatically at the beginning of the program after reset by HAL_Init() or at any time when clock is configured, by HAL_RCC_ClockConfig().
 - Source of time base is configured to generate interrupts at regular time intervals. Care must be taken if HAL_Delay() is called from a peripheral ISR process, the Tick interrupt line must have higher priority (numerically lower) than the peripheral interrupt. Otherwise the caller ISR process will be blocked.
 - functions affecting time base configurations are declared as __Weak to make override possible in case of other implementations in user file.

This section contains the following APIs:

- [**HAL_Init\(\)**](#)
- [**HAL_DelInit\(\)**](#)
- [**HAL_MspInit\(\)**](#)
- [**HAL_MspDelInit\(\)**](#)
- [**HAL_InitTick\(\)**](#)

5.1.3 HAL Control functions

This section provides functions allowing to:

- Provide a tick value in millisecond
- Provide a blocking delay in millisecond
- Suspend the time base source interrupt
- Resume the time base source interrupt
- Get the HAL API driver version
- Get the device identifier

- Get the device revision identifier
- Enable/Disable Debug module during Sleep mode
- Enable/Disable Debug module during STOP mode
- Enable/Disable Debug module during STANDBY mode

This section contains the following APIs:

- `HAL_IncTick()`
- `HAL_GetTick()`
- `HAL_Delay()`
- `HAL_SuspendTick()`
- `HAL_ResumeTick()`
- `HAL_GetHalVersion()`
- `HAL_GetREVID()`
- `HAL_GetDEVID()`
- `HAL_DBGMCU_EnableDBGSleepMode()`
- `HAL_DBGMCU_DisableDBGSleepMode()`
- `HAL_DBGMCU_EnableDBGStopMode()`
- `HAL_DBGMCU_DisableDBGStopMode()`
- `HAL_DBGMCU_EnableDBGStandbyMode()`
- `HAL_DBGMCU_DisableDBGStandbyMode()`

5.1.4 Detailed description of functions

HAL_Init

Function name	<code>HAL_StatusTypeDef HAL_Init (void)</code>
Function description	This function configures the Flash prefetch, Configures time base source, NVIC and Low level hardware.
Return values	<ul style="list-style-type: none">• HAL: status
Notes	<ul style="list-style-type: none">• This function is called at the beginning of program after reset and before the clock configuration• The time base configuration is based on MSI clock when exiting from Reset. Once done, time base tick start incrementing. In the default implementation,Systick is used as source of time base. the tick variable is incremented each 1ms in its ISR.

HAL_DeInit

Function name	<code>HAL_StatusTypeDef HAL_DeInit (void)</code>
Function description	This function de-Initializes common part of the HAL and stops the source of time base.
Return values	<ul style="list-style-type: none">• HAL: status
Notes	<ul style="list-style-type: none">• This function is optional.

HAL_MspInit

Function name	<code>void HAL_MspInit (void)</code>
Function description	Initializes the MSP.

Return values

- **None:**

HAL_MspDeInit

Function name

void HAL_MspDeInit (void)

Function description

DeInitializes the MSP.

Return values

- **None:**

HAL_InitTick

Function name

HAL_StatusTypeDef HAL_InitTick (uint32_t TickPriority)

Function description

This function configures the source of the time base.

Parameters

- **TickPriority:** Tick interrupt priority.

Return values

- **HAL:** status

Notes

- This function is called automatically at the beginning of program after reset by HAL_Init() or at any time when clock is reconfigured by HAL_RCC_ClockConfig().
- In the default implementation, SysTick timer is the source of time base. It is used to generate interrupts at regular time intervals. Care must be taken if HAL_Delay() is called from a peripheral ISR process, The the SysTick interrupt must have higher priority (numerically lower) than the peripheral interrupt. Otherwise the caller ISR process will be blocked. The function is declared as __Weak to be overwritten in case of other implementation in user file.

HAL_IncTick

Function name

void HAL_IncTick (void)

Function description

This function is called to increment a global variable "uwTick" used as application time base.

Return values

- **None:**

Notes

- In the default implementation, this variable is incremented each 1ms in Systick ISR.
- This function is declared as __weak to be overwritten in case of other implementations in user file.

HAL_Delay

Function name

void HAL_Delay (__IO uint32_t Delay)

Function description

This function provides accurate delay (in milliseconds) based on variable incremented.

Parameters

- **Delay:** specifies the delay time length, in milliseconds.

Return values

- **None:**

Notes

- In the default implementation , SysTick timer is the source of time base. It is used to generate interrupts at regular time intervals where uwTick is incremented.

- This function is declared as __weak to be overwritten in case of other implementations in user file.

HAL_GetTick

Function name	uint32_t HAL_GetTick (void)
Function description	Provides a tick value in millisecond.
Return values	<ul style="list-style-type: none">• tick: value
Notes	<ul style="list-style-type: none">• This function is declared as __weak to be overwritten in case of other implementations in user file.

HAL_SuspendTick

Function name	void HAL_SuspendTick (void)
Function description	Suspend Tick increment.
Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• In the default implementation , SysTick timer is the source of time base. It is used to generate interrupts at regular time intervals. Once HAL_SuspendTick() is called, the the SysTick interrupt will be disabled and so Tick increment is suspended.• This function is declared as __weak to be overwritten in case of other implementations in user file.

HAL_ResumeTick

Function name	void HAL_ResumeTick (void)
Function description	Resume Tick increment.
Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• In the default implementation , SysTick timer is the source of time base. It is used to generate interrupts at regular time intervals. Once HAL_ResumeTick() is called, the the SysTick interrupt will be enabled and so Tick increment is resumed.• This function is declared as __weak to be overwritten in case of other implementations in user file.

HAL_GetHalVersion

Function name	uint32_t HAL_GetHalVersion (void)
Function description	Returns the HAL revision.
Return values	<ul style="list-style-type: none">• version: 0xXYZR (8bits for each decimal, R for RC)

HAL_GetREVID

Function name	uint32_t HAL_GetREVID (void)
Function description	Returns the device revision identifier.
Return values	<ul style="list-style-type: none">• Device: revision identifier

HAL_GetDEVID

Function name **uint32_t HAL_GetDEVID (void)**

Function description Returns the device identifier.

Return values • **Device:** identifier

HAL_DBGMCU_EnableDBGSleepMode

Function name **void HAL_DBGMCU_EnableDBGSleepMode (void)**

Function description Enable the Debug Module during SLEEP mode.

Return values • **None:**

HAL_DBGMCU_DisableDBGSleepMode

Function name **void HAL_DBGMCU_DisableDBGSleepMode (void)**

Function description Disable the Debug Module during SLEEP mode.

Return values • **None:**

HAL_DBGMCU_EnableDBGStopMode

Function name **void HAL_DBGMCU_EnableDBGStopMode (void)**

Function description Enable the Debug Module during STOP mode.

Return values • **None:**

HAL_DBGMCU_DisableDBGStopMode

Function name **void HAL_DBGMCU_DisableDBGStopMode (void)**

Function description Disable the Debug Module during STOP mode.

Return values • **None:**

HAL_DBGMCU_EnableDBGStandbyMode

Function name **void HAL_DBGMCU_EnableDBGStandbyMode (void)**

Function description Enable the Debug Module during STANDBY mode.

Return values • **None:**

HAL_DBGMCU_DisableDBGStandbyMode

Function name **void HAL_DBGMCU_DisableDBGStandbyMode (void)**

Function description Disable the Debug Module during STANDBY mode.

Return values • **None:**

5.2 HAL Firmware driver defines

5.2.1 HAL

Freeze Unfreeze Peripherals in Debug mode



_HAL_DBGMCU_FREEZE_TIM2
_HAL_DBGMCU_UNFREEZE_TIM2
_HAL_DBGMCU_FREEZE_TIM3
_HAL_DBGMCU_UNFREEZE_TIM3
_HAL_DBGMCU_FREEZE_TIM4
_HAL_DBGMCU_UNFREEZE_TIM4
_HAL_DBGMCU_FREEZE_TIM5
_HAL_DBGMCU_UNFREEZE_TIM5
_HAL_DBGMCU_FREEZE_TIM6
_HAL_DBGMCU_UNFREEZE_TIM6
_HAL_DBGMCU_FREEZE_TIM7
_HAL_DBGMCU_UNFREEZE_TIM7
_HAL_DBGMCU_FREEZE_RTC
_HAL_DBGMCU_UNFREEZE_RTC
_HAL_DBGMCU_FREEZE_WWDG
_HAL_DBGMCU_UNFREEZE_WWDG
_HAL_DBGMCU_FREEZE_IWDG
_HAL_DBGMCU_UNFREEZE_IWDG
_HAL_DBGMCU_FREEZE_I2C1_TIMEOUT
_HAL_DBGMCU_UNFREEZE_I2C1_TIMEOUT
_HAL_DBGMCU_FREEZE_I2C2_TIMEOUT
_HAL_DBGMCU_UNFREEZE_I2C2_TIMEOUT
_HAL_DBGMCU_FREEZE_TIM9
_HAL_DBGMCU_UNFREEZE_TIM9
_HAL_DBGMCU_FREEZE_TIM10
_HAL_DBGMCU_UNFREEZE_TIM10
_HAL_DBGMCU_FREEZE_TIM11
_HAL_DBGMCU_UNFREEZE_TIM11

Boot Mode

SYSCFG_BOOT_MAINFLASH
SYSCFG_BOOT_SYSTEMFLASH
SYSCFG_BOOT_FSMC
SYSCFG_BOOT_SRAM

Boot Mode Configuration

_HAL_SYSCFG_REMAPMEMORY_FLASH
_HAL_SYSCFG_REMAPMEMORY_SYSTEMFLASH

`_HAL_SYSCFG_REMAPMEMORY_SRAM`
`_HAL_SYSCFG_REMAPMEMORY_FSMC`
`_HAL_SYSCFG_GET_BOOT_MODE`

Description:

- Returns the boot mode as configured by user.

Return value:

- The boot mode as configured by user. The returned value can be one of the following values:
 - SYSCFG_BOOT_MAINFLASH
 - SYSCFG_BOOT_SYSTEMFLASH
 - SYSCFG_BOOT_FSMC
(available only for STM32L151xD, STM32L152xD & STM32L162xD)
 - SYSCFG_BOOT_SRAM

USB DP line Configuration

`_HAL_SYSCFG_USBPULLUP_ENABLE`
`_HAL_SYSCFG_USBPULLUP_DISABLE`

VREFINT configuration

`_HAL_SYSCFG_VREFINT_OUT_ENABLE` The VREFINT output can be routed to any I/O in group 3:
`_HAL_SYSCFG_VREFINT_OUT_DISABLE`

6 HAL ADC Generic Driver

6.1 ADC Firmware driver registers structures

6.1.1 ADC_InitTypeDef

Data Fields

- *uint32_t ClockPrescaler*
- *uint32_t Resolution*
- *uint32_t DataAlign*
- *uint32_t ScanConvMode*
- *uint32_t EOCSelection*
- *uint32_t LowPowerAutoWait*
- *uint32_t LowPowerAutoPowerOff*
- *uint32_t ChannelsBank*
- *uint32_t ContinuousConvMode*
- *uint32_t NbrOfConversion*
- *uint32_t DiscontinuousConvMode*
- *uint32_t NbrOfDiscConversion*
- *uint32_t ExternalTrigConv*
- *uint32_t ExternalTrigConvEdge*
- *uint32_t DMAContinuousRequests*

Field Documentation

- ***uint32_t ADC_InitTypeDef::ClockPrescaler***
Select ADC clock source (asynchronous clock derived from HSI RC oscillator) and clock prescaler. This parameter can be a value of [ADC_ClockPrescaler](#) Note: In case of usage of channels on injected group, ADC frequency should be lower than AHB clock frequency /4 for resolution 12 or 10 bits, AHB clock frequency /3 for resolution 8 bits, AHB clock frequency /2 for resolution 6 bits. Note: HSI RC oscillator must be preliminarily enabled at RCC top level.
- ***uint32_t ADC_InitTypeDef::Resolution***
Configures the ADC resolution. This parameter can be a value of [ADC_Resolution](#)
- ***uint32_t ADC_InitTypeDef::DataAlign***
Specifies ADC data alignment to right (MSB on register bit 11 and LSB on register bit 0) (default setting) or to left (if regular group: MSB on register bit 15 and LSB on register bit 4, if injected group (MSB kept as signed value due to potential negative value after offset application): MSB on register bit 14 and LSB on register bit 3). This parameter can be a value of [ADC_Data_align](#)
- ***uint32_t ADC_InitTypeDef::ScanConvMode***
Configures the sequencer of regular and injected groups. This parameter can be associated to parameter 'DiscontinuousConvMode' to have main sequence subdivided in successive parts. If disabled: Conversion is performed in single mode (one channel converted, the one defined in rank 1). Parameters 'NbrOfConversion' and 'InjectedNbrOfConversion' are discarded (equivalent to set to 1). If enabled: Conversions are performed in sequence mode (multiple ranks defined by 'NbrOfConversion'/InjectedNbrOfConversion' and each channel rank). Scan direction is upward: from rank1 to rank 'n'. This parameter can be a value of [ADC_Scan_mode](#)
- ***uint32_t ADC_InitTypeDef::EOCSelection***
Specifies what EOC (End Of Conversion) flag is used for conversion by polling and interruption: end of conversion of each rank or complete sequence. This parameter

- can be a value of ***ADC_EOCSelection***. Note: For injected group, end of conversion (flag&IT) is raised only at the end of the sequence. Therefore, if end of conversion is set to end of each conversion, injected group should not be used with interruption (HAL_ADCEx_InjectedStart_IT) or polling (HAL_ADCEx_InjectedStart and HAL_ADCEx_InjectedPollForConversion). By the way, polling is still possible since driver will use an estimated timing for end of injected conversion. Note: If overrun feature is intended to be used, use ADC in mode 'interruption' (function **HAL_ADC_Start_IT()**) with parameter EOCSelection set to end of each conversion or in mode 'transfer by DMA' (function **HAL_ADC_Start_DMA()**). If overrun feature is intended to be bypassed, use ADC in mode 'polling' or 'interruption' with parameter EOCSelection must be set to end of sequence
- ***uint32_t ADC_InitTypeDef::LowPowerAutoWait***
Selects the dynamic low power Auto Delay: new conversion start only when the previous conversion (for regular group) or previous sequence (for injected group) has been treated by user software, using function **HAL_ADC_GetValue()** or **HAL_ADCEx_InjectedGetValue()**. This feature automatically adapts the speed of ADC to the speed of the system that reads the data. Moreover, this avoids risk of overrun for low frequency applications. This parameter can be a value of ***ADC_LowPowerAutoWait***. Note: Do not use with interruption or DMA (**HAL_ADC_Start_IT()**, **HAL_ADC_Start_DMA()**) since they have to clear immediately the EOC flag to free the IRQ vector sequencer. Do use with polling: 1. Start conversion with **HAL_ADC_Start()**, 2. Later on, when conversion data is needed: use **HAL_ADC_PollForConversion()** to ensure that conversion is completed and use **HAL_ADC_GetValue()** to retrieve conversion result and trig another conversion (in case of usage of injected group, use the equivalent functions **HAL_ADCExInjected_Start()**, **HAL_ADCEx_InjectedGetValue()**, ...). Note: ADC clock latency and some timing constraints depending on clock prescaler have to be taken into account: refer to reference manual (register ADC_CR2 bit DELS description).
 - ***uint32_t ADC_InitTypeDef::LowPowerAutoPowerOff***
Selects the auto-off mode: the ADC automatically powers-off after a conversion and automatically wakes-up when a new conversion is triggered (with startup time between trigger and start of sampling). This feature can be combined with automatic wait mode (parameter 'LowPowerAutoWait'). This parameter can be a value of ***ADC_LowPowerAutoPowerOff***.
 - ***uint32_t ADC_InitTypeDef::ChannelsBank***
Selects the ADC channels bank. This parameter can be a value of ***ADC_ChannelsBank***. Note: Banks availability depends on devices categories. Note: To change bank selection on the fly, without going through execution of '**HAL_ADC_Init()**', macro '**_HAL_ADC_CHANNELS_BANK()**' can be used directly.
 - ***uint32_t ADC_InitTypeDef::ContinuousConvMode***
Specifies whether the conversion is performed in single mode (one conversion) or continuous mode for regular group, after the selected trigger occurred (software start or external trigger). This parameter can be set to ENABLE or DISABLE.
 - ***uint32_t ADC_InitTypeDef::NbrOfConversion***
Specifies the number of ranks that will be converted within the regular group sequencer. To use regular group sequencer and convert several ranks, parameter 'ScanConvMode' must be enabled. This parameter must be a number between Min_Data = 1 and Max_Data = 28.
 - ***uint32_t ADC_InitTypeDef::DiscontinuousConvMode***
Specifies whether the conversions sequence of regular group is performed in Complete-sequence/Discontinuous-sequence (main sequence subdivided in successive parts). Discontinuous mode is used only if sequencer is enabled (parameter 'ScanConvMode'). If sequencer is disabled, this parameter is discarded. Discontinuous mode can be enabled only if continuous mode is disabled. If continuous

- mode is enabled, this parameter setting is discarded. This parameter can be set to ENABLE or DISABLE.
- ***uint32_t ADC_InitTypeDef::NbrOfDiscConversion***
Specifies the number of discontinuous conversions in which the main sequence of regular group (parameter NbrOfConversion) will be subdivided. If parameter 'DiscontinuousConvMode' is disabled, this parameter is discarded. This parameter must be a number between Min_Data = 1 and Max_Data = 8.
 - ***uint32_t ADC_InitTypeDef::ExternalTrigConv***
Selects the external event used to trigger the conversion start of regular group. If set to ADC_SOFTWARE_START, external triggers are disabled. If set to external trigger source, triggering is on event rising edge by default. This parameter can be a value of [**ADC_External_trigger_source-Regular**](#)
 - ***uint32_t ADC_InitTypeDef::ExternalTrigConvEdge***
Selects the external trigger edge of regular group. If trigger is set to ADC_SOFTWARE_START, this parameter is discarded. This parameter can be a value of [**ADC_External_trigger_edge-Regular**](#)
 - ***uint32_t ADC_InitTypeDef::DMAContinuousRequests***
Specifies whether the DMA requests are performed in one shot mode (DMA transfer stop when number of conversions is reached) or in Continuous mode (DMA transfer unlimited, whatever number of conversions). Note: In continuous mode, DMA must be configured in circular mode. Otherwise an overrun will be triggered when DMA buffer maximum pointer is reached. Note: This parameter must be modified when no conversion is on going on both regular and injected groups (ADC disabled, or ADC enabled without continuous mode or external trigger that could launch a conversion). This parameter can be set to ENABLE or DISABLE.

6.1.2 ADC_ChannelConfTypeDef

Data Fields

- ***uint32_t Channel***
- ***uint32_t Rank***
- ***uint32_t SamplingTime***

Field Documentation

- ***uint32_t ADC_ChannelConfTypeDef::Channel***
Specifies the channel to configure into ADC regular group. This parameter can be a value of [**ADC_channels**](#) Note: Depending on devices, some channels may not be available on package pins. Refer to device datasheet for channels availability.
Maximum number of channels by device category (without taking in account each device package constraints): STM32L1 category 1, 2: 24 channels on external pins + 3 channels on internal measurement paths (VrefInt, Temp sensor, Vcomp): Channel 0 to channel 26. STM32L1 category 3: 25 channels on external pins + 3 channels on internal measurement paths (VrefInt, Temp sensor, Vcomp): Channel 0 to channel 26, 1 additional channel in bank B. Note: OPAMP1 and OPAMP2 are connected internally but not increasing internal channels number: they are sharing ADC input with external channels ADC_IN3 and ADC_IN8. STM32L1 category 4, 5: 40 channels on external pins + 3 channels on internal measurement paths (VrefInt, Temp sensor, Vcomp): Channel 0 to channel 31, 11 additional channels in bank B. Note: OPAMP1 and OPAMP2 are connected internally but not increasing internal channels number: they are sharing ADC input with external channels ADC_IN3 and ADC_IN8. Note: In case of peripherals OPAMPx not used: 3 channels (3, 8, 13) can be configured as direct channels (fast channels). Refer to macro '[**__HAL_ADC_CHANNEL_SPEED_FAST\(\)**](#)'. Note: In case of peripheral OPAMP3 and ADC channel OPAMP3 used (OPAMP3 available on STM32L1 devices Cat.4 only): the analog switch COMP1_SW1 must be

- closed. Refer to macro: '`_HAL_OPAMP_OPAMP3OUT_CONNECT_ADC_COMP1()`'.
- **`uint32_t ADC_ChannelConfTypeDef::Rank`**
Specifies the rank in the regular group sequencer. This parameter can be a value of `ADC_regular_rank` Note: In case of need to disable a channel or change order of conversion sequencer, rank containing a previous channel setting can be overwritten by the new channel setting (or parameter number of conversions can be adjusted)
 - **`uint32_t ADC_ChannelConfTypeDef::SamplingTime`**
Sampling time value to be set for the selected channel. Unit: ADC clock cycles
Conversion time is the addition of sampling time and processing time (12 ADC clock cycles at ADC resolution 12 bits, 11 cycles at 10 bits, 9 cycles at 8 bits, 7 cycles at 6 bits). This parameter can be a value of `ADC_sampling_times` Caution: This parameter updates the parameter property of the channel, that can be used into regular and/or injected groups. If this same channel has been previously configured in the other group (regular/injected), it will be updated to last setting. Note: In case of usage of internal measurement channels (VrefInt/Vbat/TempSensor), sampling time constraints must be respected (sampling time can be adjusted in function of ADC clock frequency and sampling time setting) Refer to device datasheet for timings values, parameters TS_vrefint, TS_temp (values rough order: 4us min).

6.1.3 ADC_AnalogWDGConfTypeDef

Data Fields

- **`uint32_t WatchdogMode`**
- **`uint32_t Channel`**
- **`uint32_t ITMode`**
- **`uint32_t HighThreshold`**
- **`uint32_t LowThreshold`**
- **`uint32_t WatchdogNumber`**

Field Documentation

- **`uint32_t ADC_AnalogWDGConfTypeDef::WatchdogMode`**
Configures the ADC analog watchdog mode: single/all channels, regular/injected group. This parameter can be a value of `ADC_analog_watchdog_mode`.
- **`uint32_t ADC_AnalogWDGConfTypeDef::Channel`**
Selects which ADC channel to monitor by analog watchdog. This parameter has an effect only if watchdog mode is configured on single channel (parameter WatchdogMode) This parameter can be a value of `ADC_channels`.
- **`uint32_t ADC_AnalogWDGConfTypeDef::ITMode`**
Specifies whether the analog watchdog is configured in interrupt or polling mode. This parameter can be set to ENABLE or DISABLE
- **`uint32_t ADC_AnalogWDGConfTypeDef::HighThreshold`**
Configures the ADC analog watchdog High threshold value. This parameter must be a number between Min_Data = 0x000 and Max_Data = 0xFFFF.
- **`uint32_t ADC_AnalogWDGConfTypeDef::LowThreshold`**
Configures the ADC analog watchdog Low threshold value. This parameter must be a number between Min_Data = 0x000 and Max_Data = 0xFFFF.
- **`uint32_t ADC_AnalogWDGConfTypeDef::WatchdogNumber`**
Reserved for future use, can be set to 0

6.1.4 ADC_HandleTypeDef

Data Fields

- **`ADC_TypeDef * Instance`**

- **`ADC_InitTypeDef Init`**
- **`_IO uint32_t NbrOfConversionRank`**
- **`DMA_HandleTypeDef * DMA_Handle`**
- **`HAL_LockTypeDef Lock`**
- **`_IO uint32_t State`**
- **`_IO uint32_t ErrorCode`**

Field Documentation

- **`ADC_HandleTypeDef* ADC_HandleTypeDef::Instance`**
Register base address
- **`ADC_InitTypeDef ADC_HandleTypeDef::Init`**
ADC required parameters
- **`_IO uint32_t ADC_HandleTypeDef::NbrOfConversionRank`**
ADC conversion rank counter
- **`DMA_HandleTypeDef* ADC_HandleTypeDef::DMA_Handle`**
Pointer DMA Handler
- **`HAL_LockTypeDef ADC_HandleTypeDef::Lock`**
ADC locking object
- **`_IO uint32_t ADC_HandleTypeDef::State`**
ADC communication state (bitmap of ADC states)
- **`_IO uint32_t ADC_HandleTypeDef::ErrorCode`**
ADC Error code

6.2 ADC Firmware driver API description

6.2.1 ADC peripheral features

- 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
- Interrupt generation at the end of regular conversion, end of injected conversion, and in case of analog watchdog or overrun events.
- Single and continuous conversion modes.
- Scan mode for conversion of several channels sequentially.
- Data alignment with in-built data coherency.
- Programmable sampling time (channel wise)
- ADC conversion of regular group and injected group.
- External trigger (timer or EXTI) with configurable polarity for both regular and injected groups.
- DMA request generation for transfer of conversions data of regular group.
- ADC offset on injected channels
- ADC supply requirements: 2.4 V to 3.6 V at full speed and down to 1.8 V at slower speed.
- ADC input range: from Vref- (connected to Vssa) to Vref+ (connected to Vdda or to an external voltage reference).

6.2.2 How to use this driver

Configuration of top level parameters related to ADC

1. Enable the ADC interface
 - As prerequisite, ADC clock must be configured at RCC top level. Caution: On STM32L1, ADC clock frequency max is 16MHz (refer to device datasheet). Therefore, ADC clock prescaler must be configured in function of ADC clock source frequency to remain below this maximum frequency.

- Two clock settings are mandatory:
 - ADC clock (core clock).
 - ADC clock (conversions clock). Only one possible clock source: derived from HSI RC 16MHz oscillator (HSI). ADC is connected directly to HSI RC 16MHz oscillator. Therefore, RCC PLL setting has no impact on ADC. PLL can be disabled (".`PLL.PLLState` = `RCC_PLL_NONE`") or enabled with HSI16 as clock source (".`PLL.PLLSource` = `RCC_PLLSOURCE_HSI`") to be used as device main clock source SYSCLK. The only mandatory setting is "`HSIState` = `RCC_HSI_ON`"
 - Example: Into `HAL_ADC_MspInit()` (recommended code location) or with other device clock parameters configuration:
 - `__HAL_RCC_ADC1_CLK_ENABLE();`
 - `HAL_RCC_GetOscConfig(&RCC_OsclnItStructure);`
 - `RCC_OsclnItStructure.OscillatorType = (... | RCC OSCILLATORTYPE_HSI);`
 - `RCC_OsclnItStructure.HSIState = RCC_HSI_ON;`
 - `RCC_OsclnItStructure.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;`
 - `RCC_OsclnItStructure.PLL.PLLState = RCC_PLL_NONE;`
 - `RCC_OsclnItStructure.PLL.PLLSource = ...`
 - `RCC_OsclnItStructure.PLL...`
 - `HAL_RCC_OscConfig(&RCC_OsclnItStructure);`
 - ADC clock prescaler is configured at ADC level with parameter "ClockPrescaler" using function `HAL_ADC_Init()`.
- 2. ADC pins configuration
 - Enable the clock for the ADC GPIOs using macro `__HAL_RCC_GPIOx_CLK_ENABLE()`
 - Configure these ADC pins in analog mode using function `HAL_GPIO_Init()`
- 3. Optionally, in case of usage of ADC with interruptions:
 - Configure the NVIC for ADC using function `HAL_NVIC_EnableIRQ(ADCx_IRQn)`
 - Insert the ADC interruption handler function `HAL_ADC_IRQHandler()` into the function of corresponding ADC interruption vector `ADCx_IRQHandler()`.
- 4. Optionally, in case of usage of DMA:
 - Configure the DMA (DMA channel, mode normal or circular, ...) using function `HAL_DMA_Init()`.
 - Configure the NVIC for DMA using function `HAL_NVIC_EnableIRQ(DMAx_Channelx_IRQn)`
 - Insert the ADC interruption handler function `HAL_ADC_IRQHandler()` into the function of corresponding DMA interruption vector `DMAx_Channelx_IRQHandler()`.

Configuration of ADC, groups regular/injected, channels parameters

1. Configure the ADC parameters (resolution, data alignment, ...) and regular group parameters (conversion trigger, sequencer, ...) using function `HAL_ADC_Init()`.
2. Configure the channels for regular group parameters (channel number, channel rank into sequencer, ..., into regular group) using function `HAL_ADC_ConfigChannel()`.
3. Optionally, configure the injected group parameters (conversion trigger, sequencer, ..., of injected group) and the channels for injected group parameters (channel number, channel rank into sequencer, ..., into injected group) using function `HAL_ADCEx_InjectedConfigChannel()`.
4. Optionally, configure the analog watchdog parameters (channels monitored, thresholds, ...) using function `HAL_ADC_AnalogWDGConfig()`.
5. Optionally, for devices with several ADC instances: configure the multimode parameters using function `HAL_ADCEx_MultiModeConfigChannel()`.

Execution of ADC conversions

1. ADC driver can be used among three modes: polling, interruption, transfer by DMA.
 - ADC conversion by polling:
 - Activate the ADC peripheral and start conversions using function `HAL_ADC_Start()`
 - Wait for ADC conversion completion using function `HAL_ADC_PollForConversion()` (or for injected group: `HAL_ADCEx_InjectedPollForConversion()`)
 - Retrieve conversion results using function `HAL_ADC_GetValue()` (or for injected group: `HAL_ADCEx_InjectedGetValue()`)
 - Stop conversion and disable the ADC peripheral using function `HAL_ADC_Stop()`
 - ADC conversion by interruption:
 - Activate the ADC peripheral and start conversions using function `HAL_ADC_Start_IT()`
 - Wait for ADC conversion completion by call of function `HAL_ADC_ConvCpltCallback()` (this function must be implemented in user program) (or for injected group: `HAL_ADCEx_InjectedConvCpltCallback()`)
 - Retrieve conversion results using function `HAL_ADC_GetValue()` (or for injected group: `HAL_ADCEx_InjectedGetValue()`)
 - Stop conversion and disable the ADC peripheral using function `HAL_ADC_Stop_IT()`
 - ADC conversion with transfer by DMA:
 - Activate the ADC peripheral and start conversions using function `HAL_ADC_Start_DMA()`
 - Wait for ADC conversion completion by call of function `HAL_ADC_ConvCpltCallback()` or `HAL_ADC_ConvHalfCpltCallback()` (these functions must be implemented in user program)
 - Conversion results are automatically transferred by DMA into destination variable address.
 - Stop conversion and disable the ADC peripheral using function `HAL_ADC_Stop_DMA()`
 - For devices with several ADCs: ADC multimode conversion with transfer by DMA:
 - Activate the ADC peripheral (slave) and start conversions using function `HAL_ADC_Start()`
 - Activate the ADC peripheral (master) and start conversions using function `HAL_ADCEx_MultiModeStart_DMA()`
 - Wait for ADC conversion completion by call of function `HAL_ADC_ConvCpltCallback()` or `HAL_ADC_ConvHalfCpltCallback()` (these functions must be implemented in user program)
 - Conversion results are automatically transferred by DMA into destination variable address.
 - Stop conversion and disable the ADC peripheral (master) using function `HAL_ADCEx_MultiModeStop_DMA()`
 - Stop conversion and disable the ADC peripheral (slave) using function `HAL_ADC_Stop_IT()`



Callback functions must be implemented in user program:

- `HAL_ADC_ErrorCallback()`
- `HAL_ADC_LevelOutOfWindowCallback()` (callback of analog watchdog)
- `HAL_ADC_ConvCpltCallback()`

- HAL_ADC_ConvHalfCpltCallback
- HAL_ADCEx_InjectedConvCpltCallback()

Deinitialization of ADC

1. Disable the ADC interface
 - ADC clock can be hard reset and disabled at RCC top level.
 - Hard reset of ADC peripherals using macro `__ADCx_FORCE_RESET()`, `__ADCx_RELEASE_RESET()`.
 - ADC clock disable using the equivalent macro/functions as configuration step.
 - Example: Into `HAL_ADC_MspDelInit()` (recommended code location) or with other device clock parameters configuration:
 - `HAL_RCC_GetOscConfig(&RCC_OsclInitStructure);`
 - `RCC_OsclInitStructure.OscillatorType = RCC OSCILLATORTYPE_HSI;`
 - `RCC_OsclInitStructure.HSISState = RCC_HSI_OFF;` (if not used for system clock)
 - `HAL_RCC_OscConfig(&RCC_OsclInitStructure);`
2. ADC pins configuration
 - Disable the clock for the ADC GPIOs using macro `__HAL_RCC_GPIOx_CLK_DISABLE()`
3. Optionally, in case of usage of ADC with interruptions:
 - Disable the NVIC for ADC using function `HAL_NVIC_EnableIRQ(ADCx_IRQn)`
4. Optionally, in case of usage of DMA:
 - Deinitialize the DMA using function `HAL_DMA_Init()`.
 - Disable the NVIC for DMA using function `HAL_NVIC_EnableIRQ(DMAx_Channelx_IRQn)`

6.2.3 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize and configure the ADC.
- De-initialize the ADC.

This section contains the following APIs:

- [`HAL_ADC_Init\(\)`](#)
- [`HAL_ADC_DelInit\(\)`](#)
- [`HAL_ADC_MspInit\(\)`](#)
- [`HAL_ADC_MspDelInit\(\)`](#)

6.2.4 IO operation functions

This section provides functions allowing to:

- Start conversion of regular group.
- Stop conversion of regular group.
- Poll for conversion complete on regular group.
- Poll for conversion event.
- Get result of regular channel conversion.
- Start conversion of regular group and enable interruptions.
- Stop conversion of regular group and disable interruptions.
- Handle ADC interrupt request
- Start conversion of regular group and enable DMA transfer.

- Stop conversion of regular group and disable ADC DMA transfer.

This section contains the following APIs:

- [*HAL_ADC_Start\(\)*](#)
- [*HAL_ADC_Stop\(\)*](#)
- [*HAL_ADC_PollForConversion\(\)*](#)
- [*HAL_ADC_PollForEvent\(\)*](#)
- [*HAL_ADC_Start_IT\(\)*](#)
- [*HAL_ADC_Stop_IT\(\)*](#)
- [*HAL_ADC_Start_DMA\(\)*](#)
- [*HAL_ADC_Stop_DMA\(\)*](#)
- [*HAL_ADC_GetValue\(\)*](#)
- [*HAL_ADC_IRQHandler\(\)*](#)
- [*HAL_ADC_ConvCpltCallback\(\)*](#)
- [*HAL_ADC_ConvHalfCpltCallback\(\)*](#)
- [*HAL_ADC_LevelOutOfWindowCallback\(\)*](#)
- [*HAL_ADC_ErrorCallback\(\)*](#)

6.2.5 Peripheral Control functions

This section provides functions allowing to:

- Configure channels on regular group
- Configure the analog watchdog

This section contains the following APIs:

- [*HAL_ADC_ConfigChannel\(\)*](#)
- [*HAL_ADC_AnalogWDGConfig\(\)*](#)

6.2.6 Peripheral State and Errors functions

This subsection provides functions to get in run-time the status of the peripheral.

- Check the ADC state
- Check the ADC error code

This section contains the following APIs:

- [*HAL_ADC_GetState\(\)*](#)
- [*HAL_ADC_GetError\(\)*](#)

6.2.7 Detailed description of functions

HAL_ADC_Init

Function name	HAL_StatusTypeDef HAL_ADC_Init (ADC_HandleTypeDef * hadc)
Function description	Initializes the ADC peripheral and regular group according to parameters specified in structure "ADC_InitTypeDef".
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • As prerequisite, ADC clock must be configured at RCC top level (clock source APB2). See commented example code below that can be copied and uncommented into HAL_ADC_MspInit().

- Possibility to update parameters on the fly: This function initializes the ADC MSP (HAL_ADC_MspInit()) only when coming from ADC state reset. Following calls to this function can be used to reconfigure some parameters of ADC_InitTypeDef structure on the fly, without modifying MSP configuration. If ADC MSP has to be modified again, HAL_ADC_DeInit() must be called before HAL_ADC_Init(). The setting of these parameters is conditioned to ADC state. For parameters constraints, see comments of structure "ADC_InitTypeDef".
- This function configures the ADC within 2 scopes: scope of entire ADC and scope of regular group. For parameters details, see comments of structure "ADC_InitTypeDef".

HAL_ADC_DeInit

Function name	HAL_StatusTypeDef HAL_ADC_DeInit (ADC_HandleTypeDef * hadc)
Function description	Deinitialize the ADC peripheral registers to its default reset values.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • To not impact other ADCs, reset of common ADC registers have been left commented below. If needed, the example code can be copied and uncommented into function HAL_ADC_MspDeInit().

HAL_ADC_MspInit

Function name	void HAL_ADC_MspInit (ADC_HandleTypeDef * hadc)
Function description	Initializes the ADC MSP.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • None:

HAL_ADC_MspDeInit

Function name	void HAL_ADC_MspDeInit (ADC_HandleTypeDef * hadc)
Function description	Deinitializes the ADC MSP.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • None:

HAL_ADC_Start

Function name	HAL_StatusTypeDef HAL_ADC_Start (ADC_HandleTypeDef * hadc)
Function description	Enables ADC, starts conversion of regular group.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_ADC_Stop

Function name	HAL_StatusTypeDef HAL_ADC_Stop (ADC_HandleTypeDef * hadc)
Function description	Stop ADC conversion of regular group (and injected channels in case of auto_injection mode), disable ADC peripheral.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • HAL: status.
Notes	<ul style="list-style-type: none"> • : ADC peripheral disable is forcing stop of potential conversion on injected group. If injected group is under use, it should be preliminarily stopped using HAL_ADCEx_InjectedStop function.

HAL_ADC_PollForConversion

Function name	HAL_StatusTypeDef HAL_ADC_PollForConversion (ADC_HandleTypeDef * hadc, uint32_t Timeout)
Function description	Wait for regular group conversion to be completed.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle • Timeout: Timeout value in millisecond.
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • ADC conversion flags EOS (end of sequence) and EOC (end of conversion) are cleared by this function, with an exception: if low power feature "LowPowerAutoWait" is enabled, flags are not cleared to not interfere with this feature until data register is read using function HAL_ADC_GetValue(). • This function cannot be used in a particular setup: ADC configured in DMA mode and polling for end of each conversion (ADC init parameter "EOCSelection" set to ADC_EOC_SINGLE_CONV). In this case, DMA resets the flag EOC and polling cannot be performed on each conversion. Nevertheless, polling can still be performed on the complete sequence (ADC init parameter "EOCSelection" set to ADC_EOC_SEQ_CONV).

HAL_ADC_PollForEvent

Function name	HAL_StatusTypeDef HAL_ADC_PollForEvent (ADC_HandleTypeDef * hadc, uint32_t EventType, uint32_t Timeout)
Function description	Poll for conversion event.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle • EventType: the ADC event type. This parameter can be one of the following values: <ul style="list-style-type: none"> - ADC_AWD_EVENT: ADC Analog watchdog event. - ADC_OVR_EVENT: ADC Overrun event. • Timeout: Timeout value in millisecond.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_ADC_Start_IT

Function name **HAL_StatusTypeDef HAL_ADC_Start_IT (ADC_HandleTypeDef * hadc)**

Function description Enables ADC, starts conversion of regular group with interruption.

HAL_ADC_Stop_IT

Function name **HAL_StatusTypeDef HAL_ADC_Stop_IT (ADC_HandleTypeDef * hadc)**

Function description Stop ADC conversion of regular group (and injected group in case of auto_injection mode), disable interruption of end-of-conversion, disable ADC peripheral.

Parameters • **hadc:** ADC handle

Return values • **None:**

HAL_ADC_Start_DMA

Function name **HAL_StatusTypeDef HAL_ADC_Start_DMA (ADC_HandleTypeDef * hadc, uint32_t * pData, uint32_t Length)**

Function description Enables ADC, starts conversion of regular group and transfers result through DMA.

HAL_ADC_Stop_DMA

Function name **HAL_StatusTypeDef HAL_ADC_Stop_DMA (ADC_HandleTypeDef * hadc)**

Function description Stop ADC conversion of regular group (and injected group in case of auto_injection mode), disable ADC DMA transfer, disable ADC peripheral.

Parameters • **hadc:** ADC handle

Return values • **HAL:** status.

Notes • : ADC peripheral disable is forcing stop of potential conversion on injected group. If injected group is under use, it should be preliminarily stopped using HAL_ADCEx_InjectedStop function.

HAL_ADC_GetValue

Function name **uint32_t HAL_ADC_GetValue (ADC_HandleTypeDef * hadc)**

Function description Get ADC regular group conversion result.

Parameters • **hadc:** ADC handle

Return values • **ADC:** group regular conversion data

Notes • Reading register DR automatically clears ADC flag EOC (ADC group regular end of unitary conversion).

• This function does not clear ADC flag EOS (ADC group regular end of sequence conversion). Occurrence of flag EOS

rising: If sequencer is composed of 1 rank, flag EOS is equivalent to flag EOC. If sequencer is composed of several ranks, during the scan sequence flag EOC only is raised, at the end of the scan sequence both flags EOC and EOS are raised. To clear this flag, either use function: in programming model IT: HAL_ADC_IRQHandler(), in programming model polling: HAL_ADC_PollForConversion() or __HAL_ADC_CLEAR_FLAG(&hadc, ADC_FLAG_EOS).

HAL_ADC_IRQHandler

Function name	void HAL_ADC_IRQHandler (ADC_HandleTypeDef * hadc)
Function description	Handles ADC interrupt request.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • None:

HAL_ADC_ConvCpltCallback

Function name	void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef * hadc)
Function description	Conversion complete callback in non blocking mode.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • None:

HAL_ADC_ConvHalfCpltCallback

Function name	void HAL_ADC_ConvHalfCpltCallback (ADC_HandleTypeDef * hadc)
Function description	Conversion DMA half-transfer callback in non blocking mode.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • None:

HAL_ADC_LevelOutOfWindowCallback

Function name	void HAL_ADC_LevelOutOfWindowCallback (ADC_HandleTypeDef * hadc)
Function description	Analog watchdog callback in non blocking mode.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • None:

HAL_ADC_ErrorCallback

Function name	void HAL_ADC_ErrorCallback (ADC_HandleTypeDef * hadc)
Function description	ADC error callback in non blocking mode (ADC conversion with interruption or transfer by DMA)
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle

Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> In case of error due to overrun when using ADC with DMA transfer (HAL ADC handle parameter "ErrorCode" to state "HAL_ADC_ERROR_OVR"): Reinitialize the DMA using function "HAL_ADC_Stop_DMA()". If needed, restart a new ADC conversion using function "HAL_ADC_Start_DMA()" (this function is also clearing overrun flag)

HAL_ADC_ConfigChannel

Function name	HAL_StatusTypeDef HAL_ADC_ConfigChannel(ADC_HandleTypeDef * hadc, ADC_ChannelConfTypeDef * sConfig)
Function description	Configures the selected channel to be linked to the regular group.
Parameters	<ul style="list-style-type: none"> hadc: ADC handle sConfig: Structure of ADC channel for regular group.
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> In case of usage of internal measurement channels: Vbat/VrefInt/TempSensor. These internal paths can be disabled using function HAL_ADC_DelInit(). Possibility to update parameters on the fly: This function initializes channel into regular group, following calls to this function can be used to reconfigure some parameters of structure "ADC_ChannelConfTypeDef" on the fly, without resetting the ADC. The setting of these parameters is conditioned to ADC state. For parameters constraints, see comments of structure "ADC_ChannelConfTypeDef".

HAL_ADC_AnalogWDGConfig

Function name	HAL_StatusTypeDef HAL_ADC_AnalogWDGConfig(ADC_HandleTypeDef * hadc, ADC_AnalogWDGConfTypeDef * AnalogWDGConfig)
Function description	Configures the analog watchdog.
Parameters	<ul style="list-style-type: none"> hadc: ADC handle AnalogWDGConfig: Structure of ADC analog watchdog configuration
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> Analog watchdog thresholds can be modified while ADC conversion is on going. In this case, some constraints must be taken into account: the programmed threshold values are effective from the next ADC EOC (end of unitary conversion). Considering that registers write delay may happen due to bus activity, this might cause an uncertainty on the effective timing of the new programmed threshold values.

HAL_ADC_GetState

Function name	<code>uint32_t HAL_ADC_GetState (ADC_HandleTypeDef * hadc)</code>
Function description	return the ADC state
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_ADC_GetError

Function name	<code>uint32_t HAL_ADC_GetError (ADC_HandleTypeDef * hadc)</code>
Function description	Return the ADC error code.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • ADC: Error Code

ADC_Enable

Function name	<code>HAL_StatusTypeDef ADC_Enable (ADC_HandleTypeDef * hadc)</code>
Function description	Enable the selected ADC.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • HAL: status.
Notes	<ul style="list-style-type: none"> • Prerequisite condition to use this function: ADC must be disabled and voltage regulator must be enabled (done into <code>HAL_ADC_Init()</code>). • If low power mode AutoPowerOff is enabled, power-on/off phases are performed automatically by hardware. In this mode, this function is useless and must not be called because flag <code>ADC_FLAG_RDY</code> is not usable. Therefore, this function must be called under condition of "if (<code>hadc->Init.LowPowerAutoPowerOff != ENABLE</code>)".

ADC_ConversionStop_Disable

Function name	<code>HAL_StatusTypeDef ADC_ConversionStop_Disable (ADC_HandleTypeDef * hadc)</code>
Function description	Stop ADC conversion and disable the selected ADC.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • HAL: status.
Notes	<ul style="list-style-type: none"> • Prerequisite condition to use this function: ADC conversions must be stopped to disable the ADC.

6.3 ADC Firmware driver defines

6.3.1 ADC

ADC analog watchdog mode

ADC_ANALOGWATCHDOG_NONE
ADC_ANALOGWATCHDOG_SINGLE_REG
ADC_ANALOGWATCHDOG_SINGLE_INJEC
ADC_ANALOGWATCHDOG_SINGLE_REGINJEC
ADC_ANALOGWATCHDOG_ALL_REG
ADC_ANALOGWATCHDOG_ALL_INJEC
ADC_ANALOGWATCHDOG_ALL_REGINJEC

ADC channels

ADC_CHANNEL_0
ADC_CHANNEL_1
ADC_CHANNEL_2
ADC_CHANNEL_3
ADC_CHANNEL_4
ADC_CHANNEL_5
ADC_CHANNEL_6
ADC_CHANNEL_7
ADC_CHANNEL_8
ADC_CHANNEL_9
ADC_CHANNEL_10
ADC_CHANNEL_11
ADC_CHANNEL_12
ADC_CHANNEL_13
ADC_CHANNEL_14
ADC_CHANNEL_15
ADC_CHANNEL_16
ADC_CHANNEL_17
ADC_CHANNEL_18
ADC_CHANNEL_19
ADC_CHANNEL_20
ADC_CHANNEL_21
ADC_CHANNEL_22
ADC_CHANNEL_23
ADC_CHANNEL_24
ADC_CHANNEL_25
ADC_CHANNEL_26
ADC_CHANNEL_27

ADC_CHANNEL_28
ADC_CHANNEL_29
ADC_CHANNEL_30
ADC_CHANNEL_31
ADC_CHANNEL_TEMPSENSOR
ADC_CHANNEL_VREFINT
ADC_CHANNEL_VCOMP
ADC_CHANNEL_VOPAMP1
ADC_CHANNEL_VOPAMP2
ADC_CHANNEL_VOPAMP3

ADC ChannelsBank

ADC_CHANNELS_BANK_A
ADC_CHANNELS_BANK_B
IS_ADC_CHANNELSBANK
IS_ADC_CHANNELSBANK

ADC ClockPrescaler

ADC_CLOCK_ASYNC_DIV1	ADC asynchronous clock derived from ADC dedicated HSI without prescaler
ADC_CLOCK_ASYNC_DIV2	ADC asynchronous clock derived from ADC dedicated HSI divided by a prescaler of 2
ADC_CLOCK_ASYNC_DIV4	ADC asynchronous clock derived from ADC dedicated HSI divided by a prescaler of 4

ADC conversion group

ADC_REGULAR_GROUP
ADC_INJECTED_GROUP
ADC_REGULAR_INJECTED_GROUP

ADC Data_align

ADC_DATAALIGN_RIGHT
ADC_DATAALIGN_LEFT

ADC EOCSelection

ADC_EOC_SEQ_CONV
ADC_EOC_SINGLE_CONV

ADC Error Code

HAL_ADC_ERROR_NONE	No error
HAL_ADC_ERROR_INTERNAL	ADC IP internal error: if problem of clocking, enable/disable, erroneous state
HAL_ADC_ERROR_OVR	Overrun error
HAL_ADC_ERROR_DMA	DMA transfer error

ADC Event type

ADC_AWD_EVENT	ADC Analog watchdog event
ADC_OVR_EVENT	ADC overrun event

ADC Exported Macros

<code>_HAL_ADC_ENABLE</code>	Description: <ul style="list-style-type: none">Enable the ADC peripheral. Parameters: <ul style="list-style-type: none"><code>_HANDLE_</code>: ADC handle Return value: <ul style="list-style-type: none">None Description: <ul style="list-style-type: none">Disable the ADC peripheral. Parameters: <ul style="list-style-type: none"><code>_HANDLE_</code>: ADC handle Return value: <ul style="list-style-type: none">None Description: <ul style="list-style-type: none">Enable the ADC end of conversion interrupt. Parameters: <ul style="list-style-type: none"><code>_HANDLE_</code>: ADC handle<code>_INTERRUPT_</code>: ADC Interrupt This parameter can be any combination of the following values:<ul style="list-style-type: none">- <code>ADC_IT_EOC</code>: ADC End of Regular Conversion interrupt source- <code>ADC_IT_JEOC</code>: ADC End of Injected Conversion interrupt source- <code>ADC_IT_AWD</code>: ADC Analog watchdog interrupt source- <code>ADC_IT_OVR</code>: ADC overrun interrupt source Return value: <ul style="list-style-type: none">None Description: <ul style="list-style-type: none">Disable the ADC end of conversion interrupt. Parameters: <ul style="list-style-type: none"><code>_HANDLE_</code>: ADC handle<code>_INTERRUPT_</code>: ADC Interrupt This parameter can be any combination of the following values:<ul style="list-style-type: none">- <code>ADC_IT_EOC</code>: ADC End of Regular Conversion interrupt source- <code>ADC_IT_JEOC</code>: ADC End of Injected Conversion interrupt source
------------------------------	---

- ADC_IT_AWD: ADC Analog watchdog interrupt source
- ADC_IT_OVR: ADC overrun interrupt source

Return value:

- None

`_HAL_ADC_GET_IT_SOURCE`

Description:

- Checks if the specified ADC interrupt source is enabled or disabled.

Parameters:

- `_HANDLE_`: ADC handle
- `_INTERRUPT_`: ADC interrupt source to check
This parameter can be any combination of the following values:
 - ADC_IT_EOC: ADC End of Regular Conversion interrupt source
 - ADC_IT_JEOC: ADC End of Injected Conversion interrupt source
 - ADC_IT_AWD: ADC Analog watchdog interrupt source
 - ADC_IT_OVR: ADC overrun interrupt source

Return value:

- State: of interruption (SET or RESET)

`_HAL_ADC_GET_FLAG`

Description:

- Get the selected ADC's flag status.

Parameters:

- `_HANDLE_`: ADC handle
- `_FLAG_`: ADC flag This parameter can be any combination of the following values:
 - ADC_FLAG_STRT: ADC Regular group start flag
 - ADC_FLAG_JSTRT: ADC Injected group start flag
 - ADC_FLAG_EOC: ADC End of Regular conversion flag
 - ADC_FLAG_JEOC: ADC End of Injected conversion flag
 - ADC_FLAG_AWD: ADC Analog watchdog flag
 - ADC_FLAG_OVR: ADC overrun flag
 - ADC_FLAG_ADONS: ADC ready status flag
 - ADC_FLAG_RCNR: ADC Regular group ready status flag
 - ADC_FLAG_JCNR: ADC Injected group ready status flag

Return value:

- None

<code>__HAL_ADC_CLEAR_FLAG</code>	<p>Description:</p> <ul style="list-style-type: none"> Clear the ADC's pending flags. <p>Parameters:</p> <ul style="list-style-type: none"> <code>__HANDLE__</code>: ADC handle <code>__FLAG__</code>: ADC flag <ul style="list-style-type: none"> - <code>ADC_FLAG_STRT</code>: ADC Regular group start flag - <code>ADC_FLAG_JSTRT</code>: ADC Injected group start flag - <code>ADC_FLAG_EOC</code>: ADC End of Regular conversion flag - <code>ADC_FLAG_JEOC</code>: ADC End of Injected conversion flag - <code>ADC_FLAG_AWD</code>: ADC Analog watchdog flag - <code>ADC_FLAG_OVR</code>: ADC overrun flag - <code>ADC_FLAG_ADONS</code>: ADC ready status flag - <code>ADC_FLAG_RCNR</code>: ADC Regular group ready status flag - <code>ADC_FLAG_JCNR</code>: ADC Injected group ready status flag <p>Return value:</p> <ul style="list-style-type: none"> None
-----------------------------------	--

<code>__HAL_ADC_RESET_HANDLE_STATE</code>	<p>Description:</p> <ul style="list-style-type: none"> Reset ADC handle state. <p>Parameters:</p> <ul style="list-style-type: none"> <code>__HANDLE__</code>: ADC handle <p>Return value:</p> <ul style="list-style-type: none"> None
---	---

ADC Exported Types

<code>HAL_ADC_STATE_RESET</code>	ADC not yet initialized or disabled
<code>HAL_ADC_STATE_READY</code>	ADC peripheral ready for use
<code>HAL_ADC_STATE_BUSY_INTERNAL</code>	ADC is busy to internal process (initialization, calibration)
<code>HAL_ADC_STATE_TIMEOUT</code>	TimeOut occurrence
<code>HAL_ADC_STATE_ERROR_INTERNAL</code>	Internal error occurrence
<code>HAL_ADC_STATE_ERROR_CONFIG</code>	Configuration error occurrence
<code>HAL_ADC_STATE_ERROR_DMA</code>	DMA error occurrence
<code>HAL_ADC_STATE_REG_BUSY</code>	A conversion on group regular is ongoing or can occur (either by continuous mode, external trigger, low power auto power-on (if feature available), multimode ADC master control (if feature available))

<code>HAL_ADC_STATE_REG_EOC</code>	Conversion data available on group regular
<code>HAL_ADC_STATE_REG_OVR</code>	Overrun occurrence
<code>HAL_ADC_STATE_REG_EOSMP</code>	Not available on STM32L1 device: End Of Sampling flag raised
<code>HAL_ADC_STATE_INJ_BUSY</code>	A conversion on group injected is ongoing or can occur (either by auto-injection mode, external trigger, low power auto power-on (if feature available), multimode ADC master control (if feature available))
<code>HAL_ADC_STATE_INJ_EOC</code>	Conversion data available on group injected
<code>HAL_ADC_STATE_INJ_JQOVF</code>	Not available on STM32L1 device: Injected queue overflow occurrence
<code>HAL_ADC_STATE_AWD1</code>	Out-of-window occurrence of analog watchdog 1
<code>HAL_ADC_STATE_AWD2</code>	Not available on STM32L1 device: Out-of-window occurrence of analog watchdog 2
<code>HAL_ADC_STATE_AWD3</code>	Not available on STM32L1 device: Out-of-window occurrence of analog watchdog 3
<code>HAL_ADC_STATE_MULTIMODE_SLAVE</code>	Not available on STM32L1 device: ADC in multimode slave state, controlled by another ADC master (

ADC external trigger enable for regular group

`ADC_EXTERNALTRIGCONVEDGE_NONE`
`ADC_EXTERNALTRIGCONVEDGE_RISING`
`ADC_EXTERNALTRIGCONVEDGE_FALLING`
`ADC_EXTERNALTRIGCONVEDGE_RISINGFALLING`

ADC External trigger source Regular

`ADC_EXTERNALTRIGCONV_T2_CC3`
`ADC_EXTERNALTRIGCONV_T2_CC2`
`ADC_EXTERNALTRIGCONV_T2_TRGO`
`ADC_EXTERNALTRIGCONV_T3_CC1`
`ADC_EXTERNALTRIGCONV_T3_CC3`
`ADC_EXTERNALTRIGCONV_T3_TRGO`
`ADC_EXTERNALTRIGCONV_T4_CC4`
`ADC_EXTERNALTRIGCONV_T4_TRGO`
`ADC_EXTERNALTRIGCONV_T6_TRGO`
`ADC_EXTERNALTRIGCONV_T9_CC2`
`ADC_EXTERNALTRIGCONV_T9_TRGO`
`ADC_EXTERNALTRIGCONV_EXT_IT11`
`ADC_SOFTWARE_START`

ADC flags definition

ADC_FLAG_AWD	ADC Analog watchdog flag
ADC_FLAG_EOC	ADC End of Regular conversion flag
ADC_FLAG_JEOC	ADC End of Injected conversion flag
ADC_FLAG_JSTRT	ADC Injected group start flag
ADC_FLAG_STRT	ADC Regular group start flag
ADC_FLAG_OVR	ADC overrun flag
ADC_FLAG_ADONS	ADC ready status flag
ADC_FLAG_RCNR	ADC Regular group ready status flag
ADC_FLAG_JCNR	ADC Injected group ready status flag

ADC interrupts definition

ADC_IT_EOC	ADC End of Regular Conversion interrupt source
ADC_IT_JEOC	ADC End of Injected Conversion interrupt source
ADC_IT_AWD	ADC Analog watchdog interrupt source
ADC_IT_OVR	ADC overrun interrupt source

ADC LowPowerAutoPowerOff

ADC_AUTOPOWEROFF_DISABLE	
ADC_AUTOPOWEROFF_IDLE_PHASE	ADC power off when ADC is not converting (idle phase)
ADC_AUTOPOWEROFF_DELAY_PHASE	ADC power off when a delay is inserted between conversions (see parameter ADC_LowPowerAutoWait)
ADC_AUTOPOWEROFF_IDLE_DELAY_PHASES	ADC power off when ADC is not converting (idle phase) and when a delay is inserted between conversions

ADC LowPowerAutoWait

ADC_AUTOWAIT_DISABLE	< Note : For compatibility with other STM32 devices with ADC autowait
ADC_AUTOWAIT_UNTIL_DATA_READ	Insert a delay between ADC conversions: infinite delay, until the result of previous conversion is read
ADC_AUTOWAIT_7_APBCLOCKCYCLES	Insert a delay between ADC conversions: 7 APB clock cycles
ADC_AUTOWAIT_15_APBCLOCKCYCLES	Insert a delay between ADC conversions: 15 APB clock cycles
ADC_AUTOWAIT_31_APBCLOCKCYCLES	Insert a delay between ADC conversions: 31 APB clock cycles
ADC_AUTOWAIT_63_APBCLOCKCYCLES	Insert a delay between ADC conversions: 63 APB clock cycles
ADC_AUTOWAIT_127_APBCLOCKCYCLES	Insert a delay between ADC conversions:

127 APB clock cycles

ADC_AUTOWAIT_255_APBCLOCKCYCLES Insert a delay between ADC conversions:
255 APB clock cycles

ADC rank into regular group

ADC_REGULAR_RANK_1
ADC_REGULAR_RANK_2
ADC_REGULAR_RANK_3
ADC_REGULAR_RANK_4
ADC_REGULAR_RANK_5
ADC_REGULAR_RANK_6
ADC_REGULAR_RANK_7
ADC_REGULAR_RANK_8
ADC_REGULAR_RANK_9
ADC_REGULAR_RANK_10
ADC_REGULAR_RANK_11
ADC_REGULAR_RANK_12
ADC_REGULAR_RANK_13
ADC_REGULAR_RANK_14
ADC_REGULAR_RANK_15
ADC_REGULAR_RANK_16
ADC_REGULAR_RANK_17
ADC_REGULAR_RANK_18
ADC_REGULAR_RANK_19
ADC_REGULAR_RANK_20
ADC_REGULAR_RANK_21
ADC_REGULAR_RANK_22
ADC_REGULAR_RANK_23
ADC_REGULAR_RANK_24
ADC_REGULAR_RANK_25
ADC_REGULAR_RANK_26
ADC_REGULAR_RANK_27
ADC_REGULAR_RANK_28

ADC Resolution

ADC_RESOLUTION_12B ADC 12-bit resolution
ADC_RESOLUTION_10B ADC 10-bit resolution
ADC_RESOLUTION_8B ADC 8-bit resolution
ADC_RESOLUTION_6B ADC 6-bit resolution

ADC sampling times

ADC_SAMPLETIME_4CYCLES	Sampling time 4 ADC clock cycles
ADC_SAMPLETIME_9CYCLES	Sampling time 9 ADC clock cycles
ADC_SAMPLETIME_16CYCLES	Sampling time 16 ADC clock cycles
ADC_SAMPLETIME_24CYCLES	Sampling time 24 ADC clock cycles
ADC_SAMPLETIME_48CYCLES	Sampling time 48 ADC clock cycles
ADC_SAMPLETIME_96CYCLES	Sampling time 96 ADC clock cycles
ADC_SAMPLETIME_192CYCLES	Sampling time 192 ADC clock cycles
ADC_SAMPLETIME_384CYCLES	Sampling time 384 ADC clock cycles

ADC sampling times all channels

ADC_SAMPLETIME_ALLCHANNELS_SMPR3BIT2
ADC_SAMPLETIME_ALLCHANNELS_SMPR2BIT2
ADC_SAMPLETIME_ALLCHANNELS_SMPR1BIT2
ADC_SAMPLETIME_ALLCHANNELS_SMPR0BIT2
ADC_SAMPLETIME_ALLCHANNELS_SMPR3BIT1
ADC_SAMPLETIME_ALLCHANNELS_SMPR2BIT1
ADC_SAMPLETIME_ALLCHANNELS_SMPR1BIT1
ADC_SAMPLETIME_ALLCHANNELS_SMPR0BIT1
ADC_SAMPLETIME_ALLCHANNELS_SMPR3BIT0
ADC_SAMPLETIME_ALLCHANNELS_SMPR2BIT0
ADC_SAMPLETIME_ALLCHANNELS_SMPR1BIT0
ADC_SAMPLETIME_ALLCHANNELS_SMPR0BIT0

ADC Scan mode

ADC_SCAN_DISABLE
ADC_SCAN_ENABLE

7 HAL ADC Extension Driver

7.1 ADCEx Firmware driver registers structures

7.1.1 ADC_InjectionConfTypeDef

Data Fields

- *uint32_t InjectedChannel*
- *uint32_t InjectedRank*
- *uint32_t InjectedSamplingTime*
- *uint32_t InjectedOffset*
- *uint32_t InjectedNbrOfConversion*
- *uint32_t InjectedDiscontinuousConvMode*
- *uint32_t AutoInjectedConv*
- *uint32_t ExternalTrigInjecConv*
- *uint32_t ExternalTrigInjecConvEdge*

Field Documentation

- ***uint32_t ADC_InjectionConfTypeDef::InjectedChannel***
Selection of ADC channel to configure This parameter can be a value of **ADC_channels** Note: Depending on devices, some channels may not be available on package pins. Refer to device datasheet for channels availability.
- ***uint32_t ADC_InjectionConfTypeDef::InjectedRank***
Rank in the injected group sequencer This parameter must be a value of **ADCEx_injected_rank** Note: In case of need to disable a channel or change order of conversion sequencer, rank containing a previous channel setting can be overwritten by the new channel setting (or parameter number of conversions can be adjusted)
- ***uint32_t ADC_InjectionConfTypeDef::InjectedSamplingTime***
Sampling time value to be set for the selected channel. Unit: ADC clock cycles
Conversion time is the addition of sampling time and processing time (12 ADC clock cycles at ADC resolution 12 bits, 11 cycles at 10 bits, 9 cycles at 8 bits, 7 cycles at 6 bits). This parameter can be a value of **ADC_sampling_times** Caution: This parameter updates the parameter property of the channel, that can be used into regular and/or injected groups. If this same channel has been previously configured in the other group (regular/injected), it will be updated to last setting. Note: In case of usage of internal measurement channels (VrefInt/Vbat/TempSensor), sampling time constraints must be respected (sampling time can be adjusted in function of ADC clock frequency and sampling time setting) Refer to device datasheet for timings values, parameters TS_vrefint, TS_temp (values rough order: 4us min).
- ***uint32_t ADC_InjectionConfTypeDef::InjectedOffset***
Defines the offset to be subtracted from the raw converted data (for channels set on injected group only). Offset value must be a positive number. Depending of ADC resolution selected (12, 10, 8 or 6 bits), this parameter must be a number between Min_Data = 0x000 and Max_Data = 0xFFFF, 0x3FF, 0xFF or 0x3F respectively.
- ***uint32_t ADC_InjectionConfTypeDef::InjectedNbrOfConversion***
Specifies the number of ranks that will be converted within the injected group sequencer. To use the injected group sequencer and convert several ranks, parameter 'ScanConvMode' must be enabled. This parameter must be a number between Min_Data = 1 and Max_Data = 4. Caution: this setting impacts the entire injected group. Therefore, call of **HAL_ADCEx_InjectedConfigChannel()** to configure

- a channel on injected group can impact the configuration of other channels previously set.
- ***uint32_t ADC_InjectionConfTypeDef::InjectedDiscontinuousConvMode***
Specifies whether the conversions sequence of injected group is performed in Complete-sequence/Discontinuous-sequence (main sequence subdivided in successive parts). Discontinuous mode is used only if sequencer is enabled (parameter 'ScanConvMode'). If sequencer is disabled, this parameter is discarded. Discontinuous mode can be enabled only if continuous mode is disabled. If continuous mode is enabled, this parameter setting is discarded. This parameter can be set to ENABLE or DISABLE. Note: For injected group, number of discontinuous ranks increment is fixed to one-by-one. Caution: this setting impacts the entire injected group. Therefore, call of **HAL_ADCEx_InjectedConfigChannel()** to configure a channel on injected group can impact the configuration of other channels previously set.
 - ***uint32_t ADC_InjectionConfTypeDef::AutoInjectedConv***
Enables or disables the selected ADC automatic injected group conversion after regular one. This parameter can be set to ENABLE or DISABLE. Note: To use Automatic injected conversion, discontinuous mode must be disabled ('DiscontinuousConvMode' and 'InjectedDiscontinuousConvMode' set to DISABLE) Note: To use Automatic injected conversion, injected group external triggers must be disabled ('ExternalTrigInjecConv' set to ADC_SOFTWARE_START) Note: In case of DMA used with regular group: if DMA configured in normal mode (single shot) JAUTO will be stopped upon DMA transfer complete. To maintain JAUTO always enabled, DMA must be configured in circular mode. Caution: this setting impacts the entire injected group. Therefore, call of **HAL_ADCEx_InjectedConfigChannel()** to configure a channel on injected group can impact the configuration of other channels previously set.
 - ***uint32_t ADC_InjectionConfTypeDef::ExternalTrigInjecConv***
Selects the external event used to trigger the conversion start of injected group. If set to ADC_INJECTED_SOFTWARE_START, external triggers are disabled. If set to external trigger source, triggering is on event rising edge. This parameter can be a value of **ADCEx_External_trigger_source_Injected** Note: This parameter must be modified when ADC is disabled (before ADC start conversion or after ADC stop conversion). If ADC is enabled, this parameter setting is bypassed without error reporting (as it can be the expected behaviour in case of another parameter update on the fly) Caution: this setting impacts the entire injected group. Therefore, call of **HAL_ADCEx_InjectedConfigChannel()** to configure a channel on injected group can impact the configuration of other channels previously set.
 - ***uint32_t ADC_InjectionConfTypeDef::ExternalTrigInjecConvEdge***
Selects the external trigger edge of injected group. This parameter can be a value of **ADCEx_External_trigger_edge_Injected**. If trigger is set to ADC_INJECTED_SOFTWARE_START, this parameter is discarded. Caution: this setting impacts the entire injected group. Therefore, call of **HAL_ADCEx_InjectedConfigChannel()** to configure a channel on injected group can impact the configuration of other channels previously set.

7.2 ADCEx Firmware driver API description

7.2.1 IO operation functions

This section provides functions allowing to:

- Start conversion of injected group.
- Stop conversion of injected group.
- Poll for conversion complete on injected group.
- Get result of injected channel conversion.

- Start conversion of injected group and enable interruptions.
- Stop conversion of injected group and disable interruptions.

This section contains the following APIs:

- [*HAL_ADCEx_InjectedStart\(\)*](#)
- [*HAL_ADCEx_InjectedStop\(\)*](#)
- [*HAL_ADCEx_InjectedPollForConversion\(\)*](#)
- [*HAL_ADCEx_InjectedStart_IT\(\)*](#)
- [*HAL_ADCEx_InjectedStop_IT\(\)*](#)
- [*HAL_ADCEx_InjectedGetValue\(\)*](#)
- [*HAL_ADCEx_InjectedConvCpltCallback\(\)*](#)

7.2.2 Peripheral Control functions

This section provides functions allowing to:

- Configure channels on injected group

This section contains the following APIs:

- [*HAL_ADCEx_InjectedConfigChannel\(\)*](#)

7.2.3 Detailed description of functions

HAL_ADCEx_InjectedStart

Function name **HAL_StatusTypeDef HAL_ADCEx_InjectedStart
(ADC_HandleTypeDef * hadc)**

Function description Enables ADC, starts conversion of injected group.

Parameters • **hadc:** ADC handle

Return values • **HAL:** status

HAL_ADCEx_InjectedStop

Function name **HAL_StatusTypeDef HAL_ADCEx_InjectedStop
(ADC_HandleTypeDef * hadc)**

Function description Stop conversion of injected channels.

Parameters • **hadc:** ADC handle

Return values • **None:**

Notes

- If ADC must be disabled and if conversion is on going on regular group, function HAL_ADC_Stop must be used to stop both injected and regular groups, and disable the ADC.
- If injected group mode auto-injection is enabled, function HAL_ADC_Stop must be used.
- In case of auto-injection mode, HAL_ADC_Stop must be used.

HAL_ADCEx_InjectedPollForConversion

Function name **HAL_StatusTypeDef HAL_ADCEx_InjectedPollForConversion
(ADC_HandleTypeDef * hadc, uint32_t Timeout)**

Function description Wait for injected group conversion to be completed.

Parameters

- **hadc:** ADC handle
- **Timeout:** Timeout value in millisecond.

Return values

- **HAL:** status

HAL_ADCEx_InjectedStart_IT

Function name **HAL_StatusTypeDef HAL_ADCEx_InjectedStart_IT(ADC_HandleTypeDef * hadc)**

Function description Enables ADC, starts conversion of injected group with interruption.

HAL_ADCEx_InjectedStop_IT

Function name **HAL_StatusTypeDef HAL_ADCEx_InjectedStop_IT(ADC_HandleTypeDef * hadc)**

Function description Stop conversion of injected channels, disable interruption of end-of-conversion.

Parameters

- **hadc:** ADC handle

Return values

- **None:**

Notes

- If ADC must be disabled and if conversion is on going on regular group, function HAL_ADC_Stop must be used to stop both injected and regular groups, and disable the ADC.
- If injected group mode auto-injection is enabled, function HAL_ADC_Stop must be used.

HAL_ADCEx_InjectedGetValue

Function name **uint32_t HAL_ADCEx_InjectedGetValue (ADC_HandleTypeDef * hadc, uint32_t InjectedRank)**

Function description Get ADC injected group conversion result.

Parameters

- **hadc:** ADC handle
- **InjectedRank:** the converted ADC injected rank. This parameter can be one of the following values:
 - ADC_INJECTED_RANK_1: Injected Channel1 selected
 - ADC_INJECTED_RANK_2: Injected Channel2 selected
 - ADC_INJECTED_RANK_3: Injected Channel3 selected
 - ADC_INJECTED_RANK_4: Injected Channel4 selected

Return values

- **ADC:** group injected conversion data

Notes

- Reading register JDRx automatically clears ADC flag JEOC (ADC group injected end of unitary conversion).
- This function does not clear ADC flag JEOS (ADC group injected end of sequence conversion) Occurrence of flag JEOS rising: If sequencer is composed of 1 rank, flag JEOS is equivalent to flag JEOC. If sequencer is composed of several ranks, during the scan sequence flag JEOC only is raised, at the end of the scan sequence both flags JEOC and EOS are raised. Flag JEOS must not be cleared by this function because it would not be compliant with low power features

(feature low power auto-wait, not available on all STM32 families). To clear this flag, either use function: in programming model IT: HAL_ADC_IRQHandler(), in programming model polling: HAL_ADCEx_InjectedPollForConversion() or __HAL_ADC_CLEAR_FLAG(&hadc, ADC_FLAG_JEOS).

HAL_ADCEx_InjectedConvCpltCallback

Function name	void HAL_ADCEx_InjectedConvCpltCallback(ADC_HandleTypeDef * hadc)
Function description	Injected conversion complete callback in non blocking mode.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle
Return values	<ul style="list-style-type: none"> • None:

HAL_ADCEx_InjectedConfigChannel

Function name	HAL_StatusTypeDef HAL_ADCEx_InjectedConfigChannel(ADC_HandleTypeDef * hadc, ADC_InjectionConfTypeDef * sConfigInjected)
Function description	Configures the ADC injected group and the selected channel to be linked to the injected group.
Parameters	<ul style="list-style-type: none"> • hadc: ADC handle • sConfigInjected: Structure of ADC injected group and ADC channel for injected group.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Possibility to update parameters on the fly: This function initializes injected group, following calls to this function can be used to reconfigure some parameters of structure "ADC_InjectionConfTypeDef" on the fly, without resetting the ADC. The setting of these parameters is conditioned to ADC state: this function must be called when ADC is not under conversion.

7.3 ADCEx Firmware driver defines

7.3.1 ADCEx

ADCEx Exported Macros

<code>_HAL_ADC_CHANNELS_BANK</code>	Description:
	<ul style="list-style-type: none"> • Selection of channels bank.
	Parameters:
	<ul style="list-style-type: none"> • _HANDLE_: ADC handle • _BANK_: Bank selection. This parameter can be a value of
	Return value:
	<ul style="list-style-type: none"> • None

`_HAL_ADC_CHANNEL_SPEED_FAST` Limited to channels 3, 8, 13 and to devices category Cat.3, Cat.4, Cat.5.

`_HAL_ADC_CHANNEL_SPEED_SLOW`

ADCEx external trigger enable for injected group

`ADC_EXTERNALTRIGINJECCONV_EDGE_NONE`

`ADC_EXTERNALTRIGINJECCONV_EDGE_RISING`

`ADC_EXTERNALTRIGINJECCONV_EDGE_FALLING`

`ADC_EXTERNALTRIGINJECCONV_EDGE_RISINGFALLING`

ADCEx External trigger source Injected

`ADC_EXTERNALTRIGINJECCONV_T2_CC1`

`ADC_EXTERNALTRIGINJECCONV_T2_TRGO`

`ADC_EXTERNALTRIGINJECCONV_T3_CC4`

`ADC_EXTERNALTRIGINJECCONV_T4_TRGO`

`ADC_EXTERNALTRIGINJECCONV_T4_CC1`

`ADC_EXTERNALTRIGINJECCONV_T4_CC2`

`ADC_EXTERNALTRIGINJECCONV_T4_CC3`

`ADC_EXTERNALTRIGINJECCONV_T7_TRGO`

`ADC_EXTERNALTRIGINJECCONV_T9_CC1`

`ADC_EXTERNALTRIGINJECCONV_T9_TRGO`

`ADC_EXTERNALTRIGINJECCONV_T10_CC1`

`ADC_EXTERNALTRIGINJECCONV_EXT_IT15`

`ADC_INJECTED_SOFTWARE_START`

ADCEx injected nb conv verification

`IS_ADC_INJECTED_NB_CONV`

ADCEx rank into injected group

`ADC_INJECTED_RANK_1`

`ADC_INJECTED_RANK_2`

`ADC_INJECTED_RANK_3`

`ADC_INJECTED_RANK_4`

ADCEx Internal HAL driver Ext trig src Injected

`ADC_EXTERNALTRIGINJEC_T9_CC1`

`ADC_EXTERNALTRIGINJEC_T9_TRGO`

`ADC_EXTERNALTRIGINJEC_T2_TRGO`

`ADC_EXTERNALTRIGINJEC_T2_CC1`

`ADC_EXTERNALTRIGINJEC_T3_CC4`

`ADC_EXTERNALTRIGINJEC_T4_TRGO`

`ADC_EXTERNALTRIGINJEC_T4_CC1`

ADC_EXTERNALTRIGINJEC_T4_CC2
ADC_EXTERNALTRIGINJEC_T4_CC3
ADC_EXTERNALTRIGINJEC_T10_CC1
ADC_EXTERNALTRIGINJEC_T7_TRGO
ADC_EXTERNALTRIGINJEC_EXT_IT15

8 HAL COMP Generic Driver

8.1 COMP Firmware driver registers structures

8.1.1 COMP_InitTypeDef

Data Fields

- *uint32_t InvertingInput*
- *uint32_t NonInvertingInput*
- *uint32_t Output*
- *uint32_t Mode*
- *uint32_t WindowMode*
- *uint32_t TriggerMode*
- *uint32_t NonInvertingInputPull*

Field Documentation

- ***uint32_t COMP_InitTypeDef::InvertingInput***
Selects the inverting input of the comparator. This parameter can be a value of ***COMP_InvertingInput*** Note: Inverting input can be changed on the fly, while comparator is running. Note: This feature is available on COMP2 only. If COMP1 is selected, this parameter is discarded (On COMP1, inverting input is fixed to Vrefint).
- ***uint32_t COMP_InitTypeDef::NonInvertingInput***
Selects the non inverting input of the comparator. This parameter can be a value of ***COMPEx_NonInvertingInput***
- ***uint32_t COMP_InitTypeDef::Output***
Selects the output redirection of the comparator. This parameter can be a value of ***COMP_Output*** Note: This feature is available on COMP2 only. If COMP1 is selected, this parameter is discarded.
- ***uint32_t COMP_InitTypeDef::Mode***
Selects the operating consumption mode of the comparator to adjust the speed/consumption. This parameter can be a value of ***COMP_Mode*** Note: This feature is available on COMP2 only. If COMP1 is selected, this parameter is discarded.
- ***uint32_t COMP_InitTypeDef::WindowMode***
Selects the window mode of the 2 comparators. If enabled, non-inverting inputs of the 2 comparators are connected together and are using inputs of COMP2 only (COMP1 non-inverting input is no more accessible, even from ADC channel VCOMP). This parameter can be a value of ***COMP_WindowMode*** Note: This feature must be enabled from COMP2 instance. If COMP1 is selected, this parameter is discarded.
- ***uint32_t COMP_InitTypeDef::TriggerMode***
Selects the trigger mode of the comparator when using interruption on EXTI line (interrupt mode). This parameter can be a value of ***COMP_TriggerMode*** Note: This feature is used with function "HAL_COMP_Start_IT()". In all other functions, this parameter is discarded.
- ***uint32_t COMP_InitTypeDef::NonInvertingInputPull***
Selects the internal pulling resistor connected on non inverting input. This parameter can be a value of ***COMP_NonInvertingInputPull*** Note: To avoid extra power consumption, only one resistor should be enabled at a time. Note: This feature is available on COMP1 only. If COMP2 is selected, this parameter is discarded.

8.1.2 COMP_HandleTypeDef

Data Fields

- ***COMP_TypeDef * Instance***
- ***COMP_InitTypeDef Init***
- ***HAL_LockTypeDef Lock***
- ***__IO HAL_COMP_StateTypeDef State***

Field Documentation

- ***COMP_TypeDef* COMP_HandleTypeDef::Instance***
Register base address
- ***COMP_InitTypeDef COMP_HandleTypeDef::Init***
COMP required parameters
- ***HAL_LockTypeDef COMP_HandleTypeDef::Lock***
Locking object
- ***__IO HAL_COMP_StateTypeDef COMP_HandleTypeDef::State***
COMP communication state

8.2 COMP Firmware driver API description

8.2.1 COMP Peripheral features

The STM32L1xx device family integrates 2 analog comparators COMP1 and COMP2:

1. The non inverting input and inverting input can be set to GPIO pins. HAL COMP driver configures the Routing Interface (RI) to connect the selected I/O pins to comparator input. Caution: Comparator COMP1 and ADC cannot be used at the same time as ADC since they share the ADC switch matrix: COMP1 non-inverting input is routed through ADC switch matrix. Except if ADC is intended to measure voltage on COMP1 non-inverting input: it can be performed on ADC channel VCOMP.
2. The COMP output is available using `HAL_COMP_GetOutputLevel()`.
3. The COMP output can be redirected to embedded timers (TIM2, TIM3, TIM4, TIM10). COMP output cannot be redirected to any I/O pin.
4. The comparators COMP1 and COMP2 can be combined in window mode. In this mode, COMP2 non inverting input is used as common non-inverting input.
5. The 2 comparators have interrupt capability with wake-up from Sleep and Stop modes (through the EXTI controller):
 - COMP1 is internally connected to EXTI Line 21
 - COMP2 is internally connected to EXTI Line 22 From the corresponding IRQ handler, the right interrupt source can be retrieved with the macros
`__HAL_COMP_COMP1_EXTI_GET_FLAG()` and
`__HAL_COMP_COMP2_EXTI_GET_FLAG()`.
6. The comparators also offer the possibility to output the voltage reference (VrefInt), used on inverting inputs, on I/O pin through a buffer. To use it, refer to macro
`"__HAL_SYSCFG_VREFINT_OUT_ENABLE()"`.

8.2.2 How to use this driver

This driver provides functions to configure and program the Comparators of all STM32L1xx devices. To use the comparator, perform the following steps:

1. Initialize the COMP low level resources by implementing the `HAL_COMP_MspInit()`.
 - Configure the comparator input I/O pin using `HAL_GPIO_Init()`: - For all inputs: I/O pin in analog mode (Schmitt trigger disabled) - Possible alternate configuration, for non-inverting inputs of comparator 2: I/O pin in floating mode

(Schmitt trigger enabled). It is recommended to use analog configuration to avoid any overconsumption around VDD/2.

- Enable COMP Peripheral clock using macro
`__HAL_RCC_COMP_CLK_ENABLE()`
- If required enable the COMP interrupt (EXTI line Interrupt): enable the comparator interrupt vector using `HAL_NVIC_EnableIRQ(COMP IRQn)` and `HAL_NVIC_SetPriority(COMP IRQn, xxx, xxx)` functions.
- 2. Configure the comparator using `HAL_COMP_Init()` function:
 - Select the inverting input (COMP2 only)
 - Select the non-inverting input
 - Select the output redirection to timers (COMP2 only)
 - Select the speed mode (COMP2 only)
 - Select the window mode (related to COMP1 and COMP2, but selected by COMP2 only)
 - Select the pull-up/down resistors on non-inverting input (COMP1 only)
- 3. Enable the comparator using `HAL_COMP_Start()` or `HAL_COMP_Start_IT()` function
- 4. If needed, use `HAL_COMP_GetOutputLevel()` or `HAL_COMP_TriggerCallback()` functions to manage comparator actions (output level or events)
- 5. Disable the comparator using `HAL_COMP_Stop()` or `HAL_COMP_Stop_IT()` function
- 6. De-initialize the comparator using `HAL_COMP_DeInit()` function

8.2.3 Initialization and de-initialization functions

This section provides functions to initialize and de-initialize comparators

This section contains the following APIs:

- [`HAL_COMP_Init\(\)`](#)
- [`HAL_COMP_DeInit\(\)`](#)
- [`HAL_COMP_MspInit\(\)`](#)
- [`HAL_COMP_MspDeInit\(\)`](#)

8.2.4 IO operation functions

This subsection provides a set of functions allowing to manage the COMP start and stop actions with or without interruption on ExtI line.

This section contains the following APIs:

- [`HAL_COMP_Start\(\)`](#)
- [`HAL_COMP_Stop\(\)`](#)
- [`HAL_COMP_Start_IT\(\)`](#)
- [`HAL_COMP_Stop_IT\(\)`](#)
- [`HAL_COMP_IRQHandler\(\)`](#)

8.2.5 Peripheral Control functions

This subsection provides a set of functions allowing to control the COMP management functions: Lock status, comparator output level check, IRQ callback (in case of usage of comparator with interruption on ExtI line).

This section contains the following APIs:

- [`HAL_COMP_Lock\(\)`](#)
- [`HAL_COMP_GetOutputLevel\(\)`](#)
- [`HAL_COMP_TriggerCallback\(\)`](#)

8.2.6 Peripheral State functions

This subsection permit to get in run-time the status of the peripheral.

This section contains the following APIs:

- [**HAL_COMP_GetState\(\)**](#)

8.2.7 Detailed description of functions

HAL_COMP_Init

Function name	HAL_StatusTypeDef HAL_COMP_Init (COMP_HandleTypeDef * hcomp)
Function description	Initializes the COMP according to the specified parameters in the COMP_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • hcomp: COMP handle
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • If the selected comparator is locked, initialization can't be performed. To unlock the configuration, perform a system reset.

HAL_COMP_DeInit

Function name	HAL_StatusTypeDef HAL_COMP_DeInit (COMP_HandleTypeDef * hcomp)
Function description	Deinitializes the COMP peripheral.
Parameters	<ul style="list-style-type: none"> • hcomp: COMP handle
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • Deinitialization can't be performed if the COMP configuration is locked. To unlock the configuration, perform a system reset.

HAL_COMP_MspInit

Function name	void HAL_COMP_MspInit (COMP_HandleTypeDef * hcomp)
Function description	Initializes the COMP MSP.
Parameters	<ul style="list-style-type: none"> • hcomp: COMP handle
Return values	<ul style="list-style-type: none"> • None:

HAL_COMP_MspDeInit

Function name	void HAL_COMP_MspDeInit (COMP_HandleTypeDef * hcomp)
Function description	Deinitializes COMP MSP.
Parameters	<ul style="list-style-type: none"> • hcomp: COMP handle
Return values	<ul style="list-style-type: none"> • None:

HAL_COMP_Start

Function name **HAL_StatusTypeDef HAL_COMP_Start
(COMP_HandleTypeDef * hcomp)**

Function description Start the comparator.

Parameters • **hcomp:** COMP handle

Return values • **HAL:** status

HAL_COMP_Stop

Function name **HAL_StatusTypeDef HAL_COMP_Stop
(COMP_HandleTypeDef * hcomp)**

Function description Stop the comparator.

Parameters • **hcomp:** COMP handle

Return values • **HAL:** status

HAL_COMP_Start_IT

Function name **HAL_StatusTypeDef HAL_COMP_Start_IT
(COMP_HandleTypeDef * hcomp)**

Function description Enables the interrupt and starts the comparator.

Parameters • **hcomp:** COMP handle

Return values • **HAL:** status.

HAL_COMP_Stop_IT

Function name **HAL_StatusTypeDef HAL_COMP_Stop_IT
(COMP_HandleTypeDef * hcomp)**

Function description Disable the interrupt and Stop the comparator.

Parameters • **hcomp:** COMP handle

Return values • **HAL:** status

HAL_COMP_IRQHandler

Function name **void HAL_COMP_IRQHandler (COMP_HandleTypeDef * hcomp)**

Function description Comparator IRQ Handler.

Parameters • **hcomp:** COMP handle

Return values • **HAL:** status

HAL_COMP_Lock

Function name **HAL_StatusTypeDef HAL_COMP_Lock
(COMP_HandleTypeDef * hcomp)**

Function description Lock the selected comparator configuration.

Parameters	<ul style="list-style-type: none"> • hcomp: COMP handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_COMP_GetOutputLevel

Function name	uint32_t HAL_COMP_GetOutputLevel (COMP_HandleTypeDef * * hcomp)
Function description	Return the output level (high or low) of the selected comparator.

HAL_COMP_TriggerCallback

Function name	void HAL_COMP_TriggerCallback (COMP_HandleTypeDef * hcomp)
Function description	Comparator callback.
Parameters	<ul style="list-style-type: none"> • hcomp: COMP handle
Return values	<ul style="list-style-type: none"> • None:

HAL_COMP_GetState

Function name	HAL_COMP_StateTypeDef HAL_COMP_GetState (COMP_HandleTypeDef * hcomp)
Function description	Return the COMP state.
Parameters	<ul style="list-style-type: none"> • hcomp: : COMP handle
Return values	<ul style="list-style-type: none"> • HAL: state

8.3 COMP Firmware driver defines

8.3.1 COMP

COMP Exported Macro**_HAL_COMP_RESET_HANDLE_STATE****Description:**

- Reset COMP handle state.

Parameters:

- **_HANDLE_**: COMP handle.

Return value:

- None

_HAL_COMP_ENABLE**Description:**

- Enables the specified comparator.

Parameters:

- **_HANDLE_**: COMP handle.

Return value:

- None.

Description:

- Disables the specified comparator.

Parameters:

- __HANDLE__: COMP handle.

Return value:

- None.

Description:

- Checks whether the specified COMP flag is set or not.

Parameters:

- __HANDLE__: specifies the COMP Handle.
- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
 - COMP_FLAG_LOC_K: lock flag

Return value:

- The new state of __FLAG__ (TRUE or FALSE).

Description:

- Enable the COMP1 EXTI line rising edge trigger.

Return value:

- None

Description:

- Disable the COMP1 EXTI line rising edge trigger.

Return value:

- None

Description:

- Enable the COMP1 EXTI line falling edge trigger.

Return value:

- None

Description:

- Disable the COMP1 EXTI line falling edge trigger.

Return value:

- None

Description:

- Enable the COMP1 EXTI line rising & falling edge trigger.

Return value:

- None

Description:

- Disable the COMP1 EXTI line rising & falling edge trigger.

Return value:

- None

Description:

- Enable the COMP1 EXTI line in interrupt mode.

Return value:

- None

Description:

- Disable the COMP1 EXTI line in interrupt mode.

Return value:

- None

Description:

- Enable the COMP1 EXTI Line in event mode.

Return value:

- None

Description:

- Disable the COMP1 EXTI Line in event mode.

Return value:

- None

Description:

- Check whether the COMP1 EXTI line flag is set or not.

Return value:

- RESET: or SET

Description:

- Clear the the COMP1 EXTI flag.

Return value:

- None

Description:

- Generates a Software interrupt on COMP1 EXTI Line.

Return value:

- None

Description:

- Enable the COMP2 EXTI line rising edge trigger.

Return value:

- None

Description:

- Disable the COMP2 EXTI line rising edge trigger.

Return value:

- None

Description:

- Enable the COMP2 EXTI line falling edge trigger.

Return value:

- None

Description:

- Disable the COMP2 EXTI line falling edge trigger.

Return value:

- None

Description:

- Enable the COMP2 EXTI line rising & falling edge trigger.

Return value:

- None

Description:

- Disable the COMP2 EXTI line rising & falling edge trigger.

Return value:

- None

Description:

- Enable the COMP2 EXTI line.

Return value:

- None

Description:

- Disable the COMP2 EXTI line.

Return value:

- None

Description:

- Enable the COMP2 EXTI Line in event mode.

Return value:

- None

Description:

- Disable the COMP2 EXTI Line in event mode.

Return value:

- None

Description:

- Check whether the COMP2 EXTI line flag is set or not.

Return value:

- RESET: or SET

Description:

- Clear the the COMP2 EXTI flag.

Return value:

- None

[__HAL_COMP_COMP2_EXTI_GENERATE_SWIT](#)**Description:**

- Generates a Software interrupt on COMP1 EXTI Line.

Return value:

- None

COMP ExtiLineEvent

COMP_EXTI_LINE_COMP1 External interrupt line 21 Connected to COMP1

COMP_EXTI_LINE_COMP2 External interrupt line 22 Connected to COMP2

COMP InvertingInput

COMP_INVERTINGINPUT_IO External I/O (COMP2_INM connected to pin PB3) connected to comparator 2 inverting input

COMP_INVERTINGINPUT_VREFINT VREFINT connected to comparator 2 inverting input

COMP_INVERTINGINPUT_3_4VREFINT 3/4 VREFINT connected to comparator 2 inverting input

COMP_INVERTINGINPUT_1_2VREFINT 1/2 VREFINT connected to comparator 2 inverting input

COMP_INVERTINGINPUT_1_4VREFINT 1/4 VREFINT connected to comparator 2 inverting input

COMP_INVERTINGINPUT_DAC1 DAC_OUT1 (PA4) connected to comparator 2 inverting input

COMP_INVERTINGINPUT_DAC2 DAC2_OUT (PA5) connected to comparator 2 inverting input

IS_COMP_INVERTINGINPUT

COMP Mode

COMP_MODE_LOWSPEED Low Speed

COMP_MODE_HIGHSPEED High Speed

IS_COMP_MODE

COMP NonInvertingInputPull

COMP_NONINVERTINGINPUT_NOPULL No internal pull-up or pull-down resistor connected to comparator non inverting input

COMP_NONINVERTINGINPUT_10KPU Internal 10kOhm pull-up resistor connected to comparator non inverting input

COMP_NONINVERTINGINPUT_10KPD Internal 10kOhm pull-down resistor connected to comparator non inverting input

COMP_NONINVERTINGINPUT_400KPU Internal 400kOhm pull-up resistor connected to comparator non inverting input

COMP_NONINVERTINGINPUT_400KPD Internal 400kOhm pull-down resistor connected to comparator non inverting input

IS_COMP_NONINVERTINGINPUTPULL

COMP Output

COMP_OUTPUT_TIM2IC4	COMP2 output connected to TIM2 Input Capture 4
COMP_OUTPUT_TIM2OCREFCLR	COMP2 output connected to TIM2 OCREF Clear
COMP_OUTPUT_TIM3IC4	COMP2 output connected to TIM3 Input Capture 4
COMP_OUTPUT_TIM3OCREFCLR	COMP2 output connected to TIM3 OCREF Clear
COMP_OUTPUT_TIM4IC4	COMP2 output connected to TIM4 Input Capture 4
COMP_OUTPUT_TIM4OCREFCLR	COMP2 output connected to TIM4 OCREF Clear
COMP_OUTPUT_TIM10IC1	COMP2 output connected to TIM10 Input Capture 1
COMP_OUTPUT_NONE	COMP2 output is not connected to other peripherals

IS_COMP_OUTPUT

COMP OutputLevel

COMP_OUTPUTLEVEL_LOW	
COMP_OUTPUTLEVEL_HIGH	

COMP TriggerMode

COMP_TRIGGERMODE_NONE	No External Interrupt trigger detection
COMP_TRIGGERMODE_IT_RISING	External Interrupt Mode with Rising edge trigger detection
COMP_TRIGGERMODE_IT_FALLING	External Interrupt Mode with Falling edge trigger detection
COMP_TRIGGERMODE_IT_RISING_FALLING	External Interrupt Mode with Rising/Falling edge trigger detection

IS_COMP_TRIGGERMODE

COMP WindowMode

COMP_WINDOWMODE_DISABLE	Window mode disabled: COMP1 non-inverting input is independant
COMP_WINDOWMODE_ENABLE	Window mode enabled: COMP1 non-inverting input is no more accessible, even from ADC channel VCOMP) (connected to COMP2 non-inverting input)

IS_COMP_WINDOWMODE

9 HAL COMP Extension Driver

9.1 COMPEx Firmware driver defines

9.1.1 COMPEx

COMPEx NonInvertingInput

COMP_NONINVERTINGINPUT_PB4	I/O pin PB4 connection to COMP2 non-inverting input
COMP_NONINVERTINGINPUT_PB5	I/O pin PB5 connection to COMP2 non-inverting input
COMP_NONINVERTINGINPUT_PB6	I/O pin PB6 connection to COMP2 non-inverting input
COMP_NONINVERTINGINPUT_PB7	I/O pin PB7 connection to COMP2 non-inverting input
COMP_NONINVERTINGINPUT_NONE	In case of window mode: No I/O pin connection to COMP1 non-inverting input. Instead, connection to COMP2 non-inverting input.
COMP_NONINVERTINGINPUT_PA0	I/O pin PA0 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PA1	I/O pin PA1 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PA2	I/O pin PA2 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PA3	I/O pin PA3 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PA4	I/O pin PA4 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PA5	I/O pin PA5 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PA6	I/O pin PA5 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PA7	I/O pin PA7 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PB0	I/O pin PB0 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PB1	I/O pin PB1 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PC0	I/O pin PC0 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PC1	I/O pin PC1 connection to COMP1 non-inverting input

COMP_NONINVERTINGINPUT_PC2	I/O pin PC2 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PC3	I/O pin PC3 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PC4	I/O pin PC4 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PC5	I/O pin PC5 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PB12	I/O pin PB12 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PB13	I/O pin PB13 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PB14	I/O pin PB14 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PB15	I/O pin PB15 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PE7	I/O pin PE7 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PE8	I/O pin PE8 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PE9	I/O pin PE9 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PE10	I/O pin PE10 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PF6	I/O pin PF6 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PF7	I/O pin PF7 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PF8	I/O pin PF8 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PF9	I/O pin PF9 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_PF10	I/O pin PF10 connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_OPAMP1	OPAMP1 output connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_OPAMP2	OPAMP2 output connection to COMP1 non-inverting input
COMP_NONINVERTINGINPUT_OPAMP3	OPAMP3 output connection to COMP1 non-inverting input
IS_COMP_NONINVERTINGINPUT	

10 HAL CORTEX Generic Driver

10.1 CORTEX Firmware driver registers structures

10.1.1 MPU_Region_InitTypeDef

Data Fields

- *uint8_t Enable*
- *uint8_t Number*
- *uint32_t BaseAddress*
- *uint8_t Size*
- *uint8_t SubRegionDisable*
- *uint8_t TypeExtField*
- *uint8_t AccessPermission*
- *uint8_t DisableExec*
- *uint8_t IsShareable*
- *uint8_t IsCacheable*
- *uint8_t IsBufferable*

Field Documentation

- ***uint8_t MPU_Region_InitTypeDef::Enable***
Specifies the status of the region. This parameter can be a value of
[**CORTEX MPU Region Enable**](#)
- ***uint8_t MPU_Region_InitTypeDef::Number***
Specifies the number of the region to protect. This parameter can be a value of
[**CORTEX MPU Region Number**](#)
- ***uint32_t MPU_Region_InitTypeDef::BaseAddress***
Specifies the base address of the region to protect.
- ***uint8_t MPU_Region_InitTypeDef::Size***
Specifies the size of the region to protect. This parameter can be a value of
[**CORTEX MPU Region Size**](#)
- ***uint8_t MPU_Region_InitTypeDef::SubRegionDisable***
Specifies the number of the subregion protection to disable. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF
- ***uint8_t MPU_Region_InitTypeDef::TypeExtField***
Specifies the TEX field level. This parameter can be a value of
[**CORTEX MPU TEX Levels**](#)
- ***uint8_t MPU_Region_InitTypeDef::AccessPermission***
Specifies the region access permission type. This parameter can be a value of
[**CORTEX MPU Region Permission Attributes**](#)
- ***uint8_t MPU_Region_InitTypeDef::DisableExec***
Specifies the instruction access status. This parameter can be a value of
[**CORTEX MPU Instruction Access**](#)
- ***uint8_t MPU_Region_InitTypeDef::IsShareable***
Specifies the shareability status of the protected region. This parameter can be a value of
[**CORTEX MPU Access Shareable**](#)
- ***uint8_t MPU_Region_InitTypeDef::IsCacheable***
Specifies the cacheable status of the region protected. This parameter can be a value of
[**CORTEX MPU Access Cacheable**](#)

- ***uint8_t MPU_Region_InitTypeDef::IsBufferable***
Specifies the bufferable status of the protected region. This parameter can be a value of **CORTEX_MPUMemoryAccess_Bufferable**

10.2 CORTEX Firmware driver API description

10.2.1 Initialization and de-initialization functions

This section provide the Cortex HAL driver functions allowing to configure Interrupts Systick functionalities

This section contains the following APIs:

- ***HAL_NVIC_SetPriorityGrouping()***
- ***HAL_NVIC_SetPriority()***
- ***HAL_NVIC_EnableIRQ()***
- ***HAL_NVIC_DisableIRQ()***
- ***HAL_NVIC_SystemReset()***
- ***HAL_SYSTICK_Config()***

10.2.2 Peripheral Control functions

This subsection provides a set of functions allowing to control the CORTEX (NVIC, SYSTICK, MPU) functionalities.

This section contains the following APIs:

- ***HAL_MPU_Enable()***
- ***HAL_MPU_Disable()***
- ***HAL_MPU_ConfigRegion()***
- ***HAL_NVIC_SetPriorityGrouping()***
- ***HAL_NVIC_SetPriority()***
- ***HAL_NVIC_SetPendingIRQ()***
- ***HAL_NVIC_GetPendingIRQ()***
- ***HAL_NVIC_ClearPendingIRQ()***
- ***HAL_NVIC_GetActive()***
- ***HAL_SYSTICK_CLKSourceConfig()***
- ***HAL_SYSTICK_IRQHandler()***
- ***HAL_SYSTICK_Callback()***

10.2.3 Detailed description of functions

HAL_NVIC_SetPriorityGrouping

Function name	void HAL_NVIC_SetPriorityGrouping (uint32_t PriorityGroup)
Function description	Sets the priority grouping field (pre-emption priority and subpriority) using the required unlock sequence.
Parameters	<ul style="list-style-type: none"> • PriorityGroup: The priority grouping bits length. This parameter can be one of the following values: <ul style="list-style-type: none"> - NVIC_PRIORITYGROUP_0: 0 bits for pre-emption priority 4 bits for subpriority - NVIC_PRIORITYGROUP_1: 1 bits for pre-emption priority 3 bits for subpriority - NVIC_PRIORITYGROUP_2: 2 bits for pre-emption priority 2 bits for subpriority - NVIC_PRIORITYGROUP_3: 3 bits for pre-emption priority 1 bits for subpriority

	<ul style="list-style-type: none"> - priority 1 bits for subpriority - NVIC_PRIORITYGROUP_4: 4 bits for pre-emption - priority 0 bits for subpriority
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • When the NVIC_PriorityGroup_0 is selected, IRQ pre-emption is no more possible. The pending IRQ priority will be managed only by the subpriority.

HAL_NVIC_SetPriority

Function name	void HAL_NVIC_SetPriority (IRQn_Type IRQn, uint32_t PreemptPriority, uint32_t SubPriority)
Function description	Sets the priority of an interrupt.
Parameters	<ul style="list-style-type: none"> • IRQn: External interrupt number This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l1xx.h)) • PreemptPriority: The pre-emption priority for the IRQn channel. This parameter can be a value between 0 and 15 A lower priority value indicates a higher priority • SubPriority: the subpriority level for the IRQ channel. This parameter can be a value between 0 and 15 A lower priority value indicates a higher priority.
Return values	<ul style="list-style-type: none"> • None:

HAL_NVIC_EnableIRQ

Function name	void HAL_NVIC_EnableIRQ (IRQn_Type IRQn)
Function description	Enables a device specific interrupt in the NVIC interrupt controller.
Parameters	<ul style="list-style-type: none"> • IRQn: External interrupt number This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l1xx.h))
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • To configure interrupts priority correctly, the NVIC_PriorityGroupConfig() function should be called before.

HAL_NVIC_DisableIRQ

Function name	void HAL_NVIC_DisableIRQ (IRQn_Type IRQn)
Function description	Disables a device specific interrupt in the NVIC interrupt controller.
Parameters	<ul style="list-style-type: none"> • IRQn: External interrupt number This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l1xxxx.h))
Return values	<ul style="list-style-type: none"> • None:

HAL_NVIC_SystemReset

Function name	void HAL_NVIC_SystemReset (void)
Function description	Initiates a system reset request to reset the MCU.
Return values	<ul style="list-style-type: none"> • None:

HAL_SYSTICK_Config

Function name	uint32_t HAL_SYSTICK_Config (uint32_t TicksNumb)
Function description	Initializes the System Timer and its interrupt, and starts the System Tick Timer.
Parameters	<ul style="list-style-type: none"> • TicksNumb: Specifies the ticks Number of ticks between two interrupts.
Return values	<ul style="list-style-type: none"> • status: <ul style="list-style-type: none"> - 0 Function succeeded. - 1 Function failed.

HAL MPU_Enable

Function name	void HAL_MPU_Enable (uint32_t MPU_Control)
Function description	Enable the MPU.
Parameters	<ul style="list-style-type: none"> • MPU_Control: Specifies the control mode of the MPU during hard fault, NMI, FAULTMASK and privileged access to the default memory. This parameter can be one of the following values: <ul style="list-style-type: none"> - MPU_HFNMI_PRIVDEF_NONE - MPU_HARDFAULT_NMI - MPU_PRIVILEGED_DEFAULT - MPU_HFNMI_PRIVDEF
Return values	<ul style="list-style-type: none"> • None:

HAL_MPU_Disable

Function name	void HAL_MPU_Disable (void)
Function description	Disable the MPU.
Return values	<ul style="list-style-type: none"> • None:

HAL_MPU_ConfigRegion

Function name	void HAL_MPU_ConfigRegion (MPU_Region_InitTypeDef * MPU_Init)
Function description	Initializes and configures the Region and the memory to be protected.
Parameters	<ul style="list-style-type: none"> • MPU_Init: Pointer to a MPU_Region_InitTypeDef structure that contains the initialization and configuration information.
Return values	<ul style="list-style-type: none"> • None:

HAL_NVIC_GetPriorityGrouping

Function name	uint32_t HAL_NVIC_GetPriorityGrouping (void)
Function description	Gets the priority grouping field from the NVIC Interrupt Controller.
Return values	<ul style="list-style-type: none"> • Priority: grouping field (SCB->AIRCR [10:8] PRIGROUP field)

HAL_NVIC_GetPriority

Function name	void HAL_NVIC_GetPriority (IRQn_Type IRQn, uint32_t PriorityGroup, uint32_t * pPreemptPriority, uint32_t * pSubPriority)
Function description	Gets the priority of an interrupt.
Parameters	<ul style="list-style-type: none"> • IRQn: External interrupt number This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l1xxxx.h)) • PriorityGroup: the priority grouping bits length. This parameter can be one of the following values: <ul style="list-style-type: none"> - NVIC_PRIORITYGROUP_0: 0 bits for pre-emption priority 4 bits for subpriority - NVIC_PRIORITYGROUP_1: 1 bits for pre-emption priority 3 bits for subpriority - NVIC_PRIORITYGROUP_2: 2 bits for pre-emption priority 2 bits for subpriority - NVIC_PRIORITYGROUP_3: 3 bits for pre-emption priority 1 bits for subpriority - NVIC_PRIORITYGROUP_4: 4 bits for pre-emption priority 0 bits for subpriority • pPreemptPriority: Pointer on the Preemptive priority value (starting from 0). • pSubPriority: Pointer on the Subpriority value (starting from 0).
Return values	<ul style="list-style-type: none"> • None:

HAL_NVIC_GetPendingIRQ

Function name	uint32_t HAL_NVIC_GetPendingIRQ (IRQn_Type IRQn)
Function description	Gets Pending Interrupt (reads the pending register in the NVIC and returns the pending bit for the specified interrupt).
Parameters	<ul style="list-style-type: none"> • IRQn: External interrupt number This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l1xxxx.h))
Return values	<ul style="list-style-type: none"> • status: - 0 Interrupt status is not pending. <ul style="list-style-type: none"> - 1 Interrupt status is pending.

HAL_NVIC_SetPendingIRQ

Function name	void HAL_NVIC_SetPendingIRQ (IRQn_Type IRQn)
---------------	---

Function description	Sets Pending bit of an external interrupt.
Parameters	<ul style="list-style-type: none"> IRQn: External interrupt number This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l1xxxx.h))
Return values	<ul style="list-style-type: none"> None:

HAL_NVIC_ClearPendingIRQ

Function name	void HAL_NVIC_ClearPendingIRQ (IRQn_Type IRQn)
Function description	Clears the pending bit of an external interrupt.
Parameters	<ul style="list-style-type: none"> IRQn: External interrupt number This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l1xxxx.h))
Return values	<ul style="list-style-type: none"> None:

HAL_NVIC_GetActive

Function name	uint32_t HAL_NVIC_GetActive (IRQn_Type IRQn)
Function description	Gets active interrupt (reads the active register in NVIC and returns the active bit).
Parameters	<ul style="list-style-type: none"> IRQn: External interrupt number This parameter can be an enumerator of IRQn_Type enumeration (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32l1xxxx.h))
Return values	<ul style="list-style-type: none"> status: <ul style="list-style-type: none"> - 0 Interrupt status is not pending. - 1 Interrupt status is pending.

HAL_SYSTICK_CLKSourceConfig

Function name	void HAL_SYSTICK_CLKSourceConfig (uint32_t CLKSource)
Function description	Configures the SysTick clock source.
Parameters	<ul style="list-style-type: none"> CLKSource: specifies the SysTick clock source. This parameter can be one of the following values: <ul style="list-style-type: none"> - SYSTICK_CLKSOURCE_HCLK_DIV8: AHB clock divided by 8 selected as SysTick clock source. - SYSTICK_CLKSOURCE_HCLK: AHB clock selected as SysTick clock source.
Return values	<ul style="list-style-type: none"> None:

HAL_SYSTICK_IRQHandler

Function name	void HAL_SYSTICK_IRQHandler (void)
Function description	This function handles SYSTICK interrupt request.
Return values	<ul style="list-style-type: none"> None:

HAL_SYSTICK_Callback

Function name **void HAL_SYSTICK_Callback (void)**

Function description SYSTICK callback.

Return values • **None:**

10.3 CORTEX Firmware driver defines

10.3.1 CORTEX

CORTEX MPU Instruction Access Bufferable

MPU_ACCESS_BUFFERABLE

MPU_ACCESS_NOT_BUFFERABLE

CORTEX MPU Instruction Access Cacheable

MPU_ACCESS_CACHEABLE

MPU_ACCESS_NOT_CACHEABLE

CORTEX MPU Instruction Access Shareable

MPU_ACCESS_SHAREABLE

MPU_ACCESS_NOT_SHAREABLE

MPU HFNMI and PRIVILEGED Access control

MPU_HFNMI_PRIVDEF_NONE

MPU_HARDFAULT_NMI

MPU_PRIVILEGED_DEFAULT

MPU_HFNMI_PRIVDEF

CORTEX MPU Instruction Access

MPU_INSTRUCTION_ACCESS_ENABLE

MPU_INSTRUCTION_ACCESS_DISABLE

CORTEX MPU Region Enable

MPU_REGION_ENABLE

MPU_REGION_DISABLE

CORTEX MPU Region Number

MPU_REGION_NUMBER0

MPU_REGION_NUMBER1

MPU_REGION_NUMBER2

MPU_REGION_NUMBER3

MPU_REGION_NUMBER4

MPU_REGION_NUMBER5

MPU_REGION_NUMBER6

MPU_REGION_NUMBER7

CORTEX MPU Region Permission Attributes

MPU_REGION_NO_ACCESS
MPU_REGION_PRIV_RW
MPU_REGION_PRIV_RW_URO
MPU_REGION_FULL_ACCESS
MPU_REGION_PRIV_RO
MPU_REGION_PRIV_RO_URO

CORTEX MPU Region Size

MPU_REGION_SIZE_32B
MPU_REGION_SIZE_64B
MPU_REGION_SIZE_128B
MPU_REGION_SIZE_256B
MPU_REGION_SIZE_512B
MPU_REGION_SIZE_1KB
MPU_REGION_SIZE_2KB
MPU_REGION_SIZE_4KB
MPU_REGION_SIZE_8KB
MPU_REGION_SIZE_16KB
MPU_REGION_SIZE_32KB
MPU_REGION_SIZE_64KB
MPU_REGION_SIZE_128KB
MPU_REGION_SIZE_256KB
MPU_REGION_SIZE_512KB
MPU_REGION_SIZE_1MB
MPU_REGION_SIZE_2MB
MPU_REGION_SIZE_4MB
MPU_REGION_SIZE_8MB
MPU_REGION_SIZE_16MB
MPU_REGION_SIZE_32MB
MPU_REGION_SIZE_64MB
MPU_REGION_SIZE_128MB
MPU_REGION_SIZE_256MB
MPU_REGION_SIZE_512MB
MPU_REGION_SIZE_1GB
MPU_REGION_SIZE_2GB
MPU_REGION_SIZE_4GB

MPU TEX Levels

MPU_TEX_LEVEL0

MPU_TEX_LEVEL1

MPU_TEX_LEVEL2

CORTEX Preemption Priority Group

NVIC_PRIORITYGROUP_0 0 bits for pre-emption priority 4 bits for subpriority

NVIC_PRIORITYGROUP_1 1 bits for pre-emption priority 3 bits for subpriority

NVIC_PRIORITYGROUP_2 2 bits for pre-emption priority 2 bits for subpriority

NVIC_PRIORITYGROUP_3 3 bits for pre-emption priority 1 bits for subpriority

NVIC_PRIORITYGROUP_4 4 bits for pre-emption priority 0 bits for subpriority

CORTEX Preemption Priority Group

IS_NVIC_PRIORITY_GROUP

IS_NVIC_PREEMPTION_PRIORITY

IS_NVIC_SUB_PRIORITY

IS_NVIC_DEVICE_IRQ

CORTEX SysTick clock source

SYSTICK_CLKSOURCE_HCLK_DIV8

SYSTICK_CLKSOURCE_HCLK

11 HAL CRC Generic Driver

11.1 CRC Firmware driver registers structures

11.1.1 CRC_HandleTypeDef

Data Fields

- *CRC_TypeDef * Instance*
- *HAL_LockTypeDef Lock*
- *__IO HAL_CRC_StateTypeDef State*

Field Documentation

- ***CRC_TypeDef* CRC_HandleTypeDef::Instance***
Register base address
- ***HAL_LockTypeDef CRC_HandleTypeDef::Lock***
CRC locking object
- ***__IO HAL_CRC_StateTypeDef CRC_HandleTypeDef::State***
CRC communication state

11.2 CRC Firmware driver API description

11.2.1 How to use this driver

The CRC HAL driver can be used as follows:

1. Enable CRC AHB clock using `__HAL_RCC_CRC_CLK_ENABLE();`
2. Use `HAL_CRC_Accumulate()` function to compute the CRC value of a 32-bit data buffer using combination of the previous CRC value and the new one.
3. Use `HAL_CRC_Calculate()` function to compute the CRC Value of a new 32-bit data buffer. This function resets the CRC computation unit before starting the computation to avoid getting wrong CRC values.

11.2.2 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize the CRC according to the specified parameters in the `CRC_InitTypeDef` and create the associated handle
- Deinitialize the CRC peripheral
- Initialize the CRC MSP
- Deinitialize CRC MSP

This section contains the following APIs:

- [`HAL_CRC_Init\(\)`](#)
- [`HAL_CRC_DelInit\(\)`](#)
- [`HAL_CRC_MspInit\(\)`](#)
- [`HAL_CRC_MspDelInit\(\)`](#)

11.2.3 Peripheral Control functions

This section provides functions allowing to:

- Compute the 32-bit CRC value of 32-bit data buffer, using combination of the previous CRC value and the new one.

- Compute the 32-bit CRC value of 32-bit data buffer, independently of the previous CRC value.

This section contains the following APIs:

- [***HAL_CRC_Accumulate\(\)***](#)
- [***HAL_CRC_Calculate\(\)***](#)

11.2.4 Peripheral State functions

This subsection permits to get in run-time the status of the peripheral and the data flow.

This section contains the following APIs:

- [***HAL_CRC_GetState\(\)***](#)
- [***HAL_CRC_Accumulate\(\)***](#)
- [***HAL_CRC_Calculate\(\)***](#)

11.2.5 Detailed description of functions

[*HAL_CRC_Init*](#)

Function name	<code>HAL_StatusTypeDef HAL_CRC_Init (CRC_HandleTypeDef * hcrc)</code>
Function description	Initializes the CRC according to the specified parameters in the CRC_InitTypeDef and creates the associated handle.
Parameters	<ul style="list-style-type: none"> • hcrc: pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC
Return values	<ul style="list-style-type: none"> • HAL: status

[*HAL_CRC_DeInit*](#)

Function name	<code>HAL_StatusTypeDef HAL_CRC_DeInit (CRC_HandleTypeDef * hcrc)</code>
Function description	DeInitializes the CRC peripheral.
Parameters	<ul style="list-style-type: none"> • hcrc: pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC
Return values	<ul style="list-style-type: none"> • HAL: status

[*HAL_CRC_MspInit*](#)

Function name	<code>void HAL_CRC_MspInit (CRC_HandleTypeDef * hcrc)</code>
Function description	Initializes the CRC MSP.
Parameters	<ul style="list-style-type: none"> • hcrc: pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC
Return values	<ul style="list-style-type: none"> • None:

[*HAL_CRC_MspDeInit*](#)

Function name	<code>void HAL_CRC_MspDeInit (CRC_HandleTypeDef * hcrc)</code>
Function description	DeInitializes the CRC MSP.

Parameters	<ul style="list-style-type: none"> hcrc: pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC
Return values	<ul style="list-style-type: none"> None:

HAL_CRC_Accumulate

Function name	<code>uint32_t HAL_CRC_Accumulate (CRC_HandleTypeDef * hcrc, uint32_t pBuffer, uint32_t BufferLength)</code>
Function description	Computes the 32-bit CRC of 32-bit data buffer using combination of the previous CRC value and the new one.
Parameters	<ul style="list-style-type: none"> hcrc: pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC pBuffer: pointer to the buffer containing the data to be computed BufferLength: length of the buffer to be computed (defined in word, 4 bytes)
Return values	<ul style="list-style-type: none"> 32-bit: CRC

HAL_CRC_Calculate

Function name	<code>uint32_t HAL_CRC_Calculate (CRC_HandleTypeDef * hcrc, uint32_t pBuffer, uint32_t BufferLength)</code>
Function description	Computes the 32-bit CRC of 32-bit data buffer independently of the previous CRC value.
Parameters	<ul style="list-style-type: none"> hcrc: pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC pBuffer: Pointer to the buffer containing the data to be computed BufferLength: Length of the buffer to be computed (defined in word, 4 bytes)
Return values	<ul style="list-style-type: none"> 32-bit: CRC

HAL_CRC_GetState

Function name	<code>HAL_CRC_StateTypeDef HAL_CRC_GetState (CRC_HandleTypeDef * hcrc)</code>
Function description	Returns the CRC state.
Parameters	<ul style="list-style-type: none"> hcrc: pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC
Return values	<ul style="list-style-type: none"> HAL: state

11.3 CRC Firmware driver defines

11.3.1 CRC

CRC Exported Macros

`_HAL_CRC_RESET_HANDLE_STATE` **Description:**

- Reset CRC handle state.

Parameters:

- `__HANDLE__`: CRC handle

Return value:

- None

`__HAL_CRC_DR_RESET`

Description:

- Resets CRC Data Register.

Parameters:

- `__HANDLE__`: CRC handle

Return value:

- None

`__HAL_CRC_SET_IDR`

Description:

- Stores a 8-bit data in the Independent Data(ID) register.

Parameters:

- `__HANDLE__`: CRC handle
- `__VALUE__`: 8-bit value to be stored in the ID register

Return value:

- None

`__HAL_CRC_GET_IDR`

Description:

- Returns the 8-bit data stored in the Independent Data(ID) register.

Parameters:

- `__HANDLE__`: CRC handle

Return value:

- 8-bit: value of the ID register

12 HAL CRYP Generic Driver

12.1 CRYP Firmware driver registers structures

12.1.1 CRYP_InitTypeDef

Data Fields

- *uint32_t DataType*
- *uint8_t * pKey*
- *uint8_t * pInitVect*

Field Documentation

- ***uint32_t CRYP_InitTypeDef::DataType***
32-bit data, 16-bit data, 8-bit data or 1-bit string. This parameter can be a value of [CRYP_Data_Type](#)
- ***uint8_t* CRYP_InitTypeDef::pKey***
The key used for encryption/decryption
- ***uint8_t* CRYP_InitTypeDef::pInitVect***
The initialization vector used also as initialization counter in CTR mode

12.1.2 CRYP_HandleTypeDefDef

Data Fields

- *AES_TypeDef * Instance*
- *CRYP_InitTypeDef Init*
- *uint8_t * pCrypInBuffPtr*
- *uint8_t * pCrypOutBuffPtr*
- *_IO uint16_t CrypInCount*
- *_IO uint16_t CrypOutCount*
- *HAL_StatusTypeDef Status*
- *HAL_PhaseTypeDef Phase*
- *DMA_HandleTypeDef * hdmain*
- *DMA_HandleTypeDef * hdmaout*
- *HAL_LockTypeDef Lock*
- *_IO HAL_CRYP_STATETypeDef State*

Field Documentation

- ***AES_TypeDef* CRYP_HandleTypeDefDef::Instance***
Register base address
- ***CRYP_InitTypeDef CRYP_HandleTypeDefDef::Init***
CRYP required parameters
- ***uint8_t* CRYP_HandleTypeDefDef::pCrypInBuffPtr***
Pointer to CRYP processing (encryption, decryption,...) buffer
- ***uint8_t* CRYP_HandleTypeDefDef::pCrypOutBuffPtr***
Pointer to CRYP processing (encryption, decryption,...) buffer
- ***_IO uint16_t CRYP_HandleTypeDefDef::CrypInCount***
Counter of inputed data
- ***_IO uint16_t CRYP_HandleTypeDefDef::CrypOutCount***
Counter of outputed data
- ***HAL_StatusTypeDef CRYP_HandleTypeDefDef::Status***
CRYP peripheral status

- ***HAL_PhaseTypeDef CRYPT_HandleTypeDef::Phase***
CRYPT peripheral phase
- ***DMA_HandleTypeDef* CRYPT_HandleTypeDef::hdmain***
CRYPT In DMA handle parameters
- ***DMA_HandleTypeDef* CRYPT_HandleTypeDef::hdmaout***
CRYPT Out DMA handle parameters
- ***HAL_LockTypeDef CRYPT_HandleTypeDef::Lock***
CRYPT locking object
- ***_IO HAL_CRYPT_STATETypeDef CRYPT_HandleTypeDef::State***
CRYPT peripheral state

12.2 CRYPT Firmware driver API description

12.2.1 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize the CRYPT according to the specified parameters in the CRYPT_InitTypeDef and creates the associated handle
- Deinitialize the CRYPT peripheral
- Initialize the CRYPT MSP
- Deinitialize CRYPT MSP

This section contains the following APIs:

- [***HAL_CRYPT_Init\(\)***](#)
- [***HAL_CRYPT_DelInit\(\)***](#)
- [***HAL_CRYPT_MspInit\(\)***](#)
- [***HAL_CRYPT_MspDelInit\(\)***](#)

12.2.2 AES processing functions

This section provides functions allowing to:

- Encrypt plaintext using AES algorithm in different chaining modes
- Decrypt ciphertext using AES algorithm in different chaining modes

Three processing functions are available:

- Polling mode
- Interrupt mode
- DMA mode

This section contains the following APIs:

- [***HAL_CRYPT_AESECB_Encrypt\(\)***](#)
- [***HAL_CRYPT_AESCBC_Encrypt\(\)***](#)
- [***HAL_CRYPT_AESCTR_Encrypt\(\)***](#)
- [***HAL_CRYPT_AESECB_Decrypt\(\)***](#)
- [***HAL_CRYPT_AESCBC_Decrypt\(\)***](#)
- [***HAL_CRYPT_AESCTR_Decrypt\(\)***](#)
- [***HAL_CRYPT_AESECB_Encrypt_IT\(\)***](#)
- [***HAL_CRYPT_AESCBC_Encrypt_IT\(\)***](#)
- [***HAL_CRYPT_AESCTR_Encrypt_IT\(\)***](#)
- [***HAL_CRYPT_AESECB_Decrypt_IT\(\)***](#)
- [***HAL_CRYPT_AESCBC_Decrypt_IT\(\)***](#)
- [***HAL_CRYPT_AESCTR_Decrypt_IT\(\)***](#)
- [***HAL_CRYPT_AESECB_Encrypt_DMA\(\)***](#)

- [*HAL_CRYP_AESCBC_Encrypt_DMA\(\)*](#)
- [*HAL_CRYP_AESCTR_Encrypt_DMA\(\)*](#)
- [*HAL_CRYP_AESECB_Decrypt_DMA\(\)*](#)
- [*HAL_CRYP_AESCBC_Decrypt_DMA\(\)*](#)
- [*HAL_CRYP_AESCTR_Decrypt_DMA\(\)*](#)

12.2.3 DMA callback functions

This section provides DMA callback functions:

- DMA Input data transfer complete
- DMA Output data transfer complete
- DMA error

This section contains the following APIs:

- [*HAL_CRYP_ErrorCallback\(\)*](#)
- [*HAL_CRYP_InCpltCallback\(\)*](#)
- [*HAL_CRYP_OutCpltCallback\(\)*](#)

12.2.4 CRYP IRQ handler management

This section provides CRYP IRQ handler function.

This section contains the following APIs:

- [*HAL_CRYP_IRQHandler\(\)*](#)

12.2.5 Peripheral State functions

This subsection permits to get in run-time the status of the peripheral.

This section contains the following APIs:

- [*HAL_CRYP_GetState\(\)*](#)

12.2.6 Detailed description of functions

HAL_CRYP_Init

Function name	HAL_StatusTypeDef HAL_CRYP_Init (CRYP_HandleTypeDef * hcryp)
Function description	Initializes the CRYP according to the specified parameters in the CRYP_InitTypeDef and creates the associated handle.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_DeInit

Function name	HAL_StatusTypeDef HAL_CRYP_DeInit (CRYP_HandleTypeDef * hcryp)
Function description	Deinitializes the CRYP peripheral.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_MspInit

Function name	void HAL_CRYP_MspInit (CRYP_HandleTypeDef * hcryp)
Function description	Initializes the CRYP MSP.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> • None:

HAL_CRYP_MspDeInit

Function name	void HAL_CRYP_MspDeInit (CRYP_HandleTypeDef * hcryp)
Function description	Deinitializes CRYP MSP.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> • None:

HAL_CRYP_AESECB_Encrypt

Function name	HAL_StatusTypeDef HAL_CRYP_AESECB_Encrypt (CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData, uint32_t Timeout)
Function description	Initializes the CRYP peripheral in AES ECB encryption mode then encrypt pPlainData.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16. • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Timeout: Specify Timeout value
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESECB_Decrypt

Function name	HAL_StatusTypeDef HAL_CRYP_AESECB_Decrypt (CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData, uint16_t Size, uint8_t * pPlainData, uint32_t Timeout)
Function description	Initializes the CRYP peripheral in AES ECB decryption mode then decrypted pCypherData.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16. • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Timeout: Specify Timeout value
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESCBC_Encrypt

Function name	HAL_StatusTypeDef HAL_CRYP_AESCBC_Encrypt (CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData, uint32_t Timeout)
Function description	Initializes the CRYP peripheral in AES CBC encryption mode then encrypt pPlainData.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16. • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Timeout: Specify Timeout value
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESCBC_Decrypt

Function name	HAL_StatusTypeDef HAL_CRYP_AESCBC_Decrypt (CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData, uint16_t Size, uint8_t * pPlainData, uint32_t Timeout)
Function description	Initializes the CRYP peripheral in AES ECB decryption mode then decrypted pCypherData.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16. • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Timeout: Specify Timeout value
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESCTR_Encrypt

Function name	HAL_StatusTypeDef HAL_CRYP_AESCTR_Encrypt (CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData, uint32_t Timeout)
Function description	Initializes the CRYP peripheral in AES CTR encryption mode then encrypt pPlainData.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16. • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Timeout: Specify Timeout value
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESCTR_Decrypt

Function name	HAL_StatusTypeDef HAL_CRYP_AESCTR_Decrypt (CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData, uint16_t Size, uint8_t * pPlainData, uint32_t Timeout)
Function description	Initializes the CRYP peripheral in AES CTR decryption mode then decrypted pCypherData.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16. • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Timeout: Specify Timeout value
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESECB_Encrypt_IT

Function name	HAL_StatusTypeDef HAL_CRYP_AESECB_Encrypt_IT (CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData)
Function description	Initializes the CRYP peripheral in AES ECB encryption mode using Interrupt.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16 bytes • pCypherData: Pointer to the ciphertext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESCBC_Encrypt_IT

Function name	HAL_StatusTypeDef HAL_CRYP_AESCBC_Encrypt_IT (CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData)
Function description	Initializes the CRYP peripheral in AES CBC encryption mode using Interrupt.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16 bytes • pCypherData: Pointer to the ciphertext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESCTR_Encrypt_IT

Function name	HAL_StatusTypeDef HAL_CRYP_AESCTR_Encrypt_IT (CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData)
Function description	Initializes the CRYP peripheral in AES CTR encryption mode using Interrupt.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16 bytes • pCypherData: Pointer to the ciphertext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESECB_Decrypt_IT

Function name	HAL_StatusTypeDef HAL_CRYP_AESECB_Decrypt_IT (CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData, uint16_t Size, uint8_t * pPlainData)
Function description	Initializes the CRYP peripheral in AES ECB decryption mode using Interrupt.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16. • pPlainData: Pointer to the plaintext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESCTR_Decrypt_IT

Function name	HAL_StatusTypeDef HAL_CRYP_AESCTR_Decrypt_IT (CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData, uint16_t Size, uint8_t * pPlainData)
Function description	Initializes the CRYP peripheral in AES CTR decryption mode using Interrupt.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16 • pPlainData: Pointer to the plaintext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_CRYP_AESCBC_Decrypt_IT

Function name	HAL_StatusTypeDef HAL_CRYP_AESCBC_Decrypt_IT (CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData,
---------------	---

`uint16_t Size, uint8_t * pPlainData)`

Function description	Initializes the CRYP peripheral in AES CBC decryption mode using IT.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16 • pPlainData: Pointer to the plaintext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> • HAL: status

`HAL_CRYP_AESECB_Encrypt_DMA`

Function name	<code>HAL_StatusTypeDef HAL_CRYP_AESECB_Encrypt_DMA(CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData)</code>
Function description	Initializes the CRYP peripheral in AES ECB encryption mode using DMA.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pPlainData: Pointer to the plaintext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16 bytes • pCypherData: Pointer to the ciphertext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> • HAL: status

`HAL_CRYP_AESECB_Decrypt_DMA`

Function name	<code>HAL_StatusTypeDef HAL_CRYP_AESECB_Decrypt_DMA(CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData, uint16_t Size, uint8_t * pPlainData)</code>
Function description	Initializes the CRYP peripheral in AES ECB decryption mode using DMA.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module • pCypherData: Pointer to the ciphertext buffer (aligned on u32) • Size: Length of the plaintext buffer, must be a multiple of 16 bytes • pPlainData: Pointer to the plaintext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> • HAL: status

`HAL_CRYP_AESCBC_Encrypt_DMA`

Function name	<code>HAL_StatusTypeDef HAL_CRYP_AESCBC_Encrypt_DMA(CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData)</code>
---------------	---

Function description	Initializes the CRYP peripheral in AES CBC encryption mode using DMA.
Parameters	<ul style="list-style-type: none"> hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module pPlainData: Pointer to the plaintext buffer (aligned on u32) Size: Length of the plaintext buffer, must be a multiple of 16. pCypherData: Pointer to the ciphertext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> HAL: status

HAL_CRYP_AESCBC_Decrypt_DMA

Function name	HAL_StatusTypeDef HAL_CRYP_AESCBC_Decrypt_DMA (CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData, uint16_t Size, uint8_t * pPlainData)
Function description	Initializes the CRYP peripheral in AES CBC encryption mode using DMA.
Parameters	<ul style="list-style-type: none"> hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module pCypherData: Pointer to the ciphertext buffer (aligned on u32) Size: Length of the plaintext buffer, must be a multiple of 16 bytes pPlainData: Pointer to the plaintext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> HAL: status

HAL_CRYP_AESCTR_Encrypt_DMA

Function name	HAL_StatusTypeDef HAL_CRYP_AESCTR_Encrypt_DMA (CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t Size, uint8_t * pCypherData)
Function description	Initializes the CRYP peripheral in AES CTR encryption mode using DMA.
Parameters	<ul style="list-style-type: none"> hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module pPlainData: Pointer to the plaintext buffer (aligned on u32) Size: Length of the plaintext buffer, must be a multiple of 16. pCypherData: Pointer to the ciphertext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> HAL: status

HAL_CRYP_AESCTR_Decrypt_DMA

Function name	HAL_StatusTypeDef HAL_CRYP_AESCTR_Decrypt_DMA (CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData, uint16_t Size, uint8_t * pPlainData)
Function description	Initializes the CRYP peripheral in AES CTR decryption mode using DMA.

Parameters	<ul style="list-style-type: none"> hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module pCypherData: Pointer to the ciphertext buffer (aligned on u32) Size: Length of the plaintext buffer, must be a multiple of 16 pPlainData: Pointer to the plaintext buffer (aligned on u32)
Return values	<ul style="list-style-type: none"> HAL: status

HAL_CRYP_InCpltCallback

Function name	void HAL_CRYP_InCpltCallback (CRYP_HandleTypeDef * hcryp)
Function description	Input transfer completed callback.
Parameters	<ul style="list-style-type: none"> hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> None:

HAL_CRYP_OutCpltCallback

Function name	void HAL_CRYP_OutCpltCallback (CRYP_HandleTypeDef * hcryp)
Function description	Output transfer completed callback.
Parameters	<ul style="list-style-type: none"> hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> None:

HAL_CRYP_ErrorCallback

Function name	void HAL_CRYP_ErrorCallback (CRYP_HandleTypeDef * hcryp)
Function description	CRYP error callback.
Parameters	<ul style="list-style-type: none"> hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> None:

HAL_CRYP_IRQHandler

Function name	void HAL_CRYP_IRQHandler (CRYP_HandleTypeDef * hcryp)
Function description	This function handles CRYP interrupt request.
Parameters	<ul style="list-style-type: none"> hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> None:

HAL_CRYP_GetState

Function name	HAL_CRYP_STATETypeDef HAL_CRYP_GetState (CRYP_HandleTypeDef * hcryp)
---------------	---

Function description	Returns the CRYP state.
Parameters	<ul style="list-style-type: none"> • hcryp: pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module
Return values	<ul style="list-style-type: none"> • HAL: state

12.3 CRYP Firmware driver defines

12.3.1 CRYP

AES Clear Flags

CRYP_CLEARFLAG_CCF	Computation Complete Flag Clear
CRYP_CLEARFLAG_RDERR	Read Error Clear
CRYP_CLEARFLAG_WRERR	Write Error Clear

AES Flags

CRYP_FLAG_CCF	Computation Complete Flag
CRYP_FLAG_RDERR	Read Error Flag
CRYP_FLAG_WRERR	Write Error Flag

AES Interrupts

CRYP_IT_CC	Computation Complete interrupt
CRYP_IT_ERR	Error interrupt

CRYP Algo Mode Direction

CRYP_CR_ALGOMODE_DIRECTION	
CRYP_CR_ALGOMODE_AES_ECB_ENCRYPT	
CRYP_CR_ALGOMODE_AES_ECB_KEYDERDECRYPT	
CRYP_CR_ALGOMODE_AES_CBC_ENCRYPT	
CRYP_CR_ALGOMODE_AES_CBC_KEYDERDECRYPT	
CRYP_CR_ALGOMODE_AES_CTR_ENCRYPT	
CRYP_CR_ALGOMODE_AES_CTR_DECRYPT	

CRYP Data Type

CRYP_DATATYPE_32B	
CRYP_DATATYPE_16B	
CRYP_DATATYPE_8B	
CRYP_DATATYPE_1B	
IS_CRYP_DATATYPE	

CRYP Exported Macros

<code>_HAL_CRYP_RESET_HANDLE_STATE</code>	Description:
	<ul style="list-style-type: none"> • Reset CRYP handle state.
	Parameters:
	<ul style="list-style-type: none"> • <code>_HANDLE_</code>: specifies the CRYP

handle.

Return value:

- None

`__HAL_CRYP_ENABLE`

Description:

- Enable/Disable the CRYP peripheral.

Parameters:

- `__HANDLE__`: specifies the CRYP handle.

Return value:

- None

`__HAL_CRYP_DISABLE`

`__HAL_CRYP_SET_MODE`

Description:

- Set the algorithm mode: AES-ECB, AES-CBC, AES-CTR, DES-ECB, DES-CBC,...

Parameters:

- `__HANDLE__`: specifies the CRYP handle.
- `__MODE__`: The algorithm mode.

Return value:

- None

`__HAL_CRYP_GET_FLAG`

Description:

- Check whether the specified CRYP flag is set or not.

Parameters:

- `__HANDLE__`: specifies the CRYP handle.
- `__FLAG__`: specifies the flag to check. This parameter can be one of the following values:
 - `CRYP_FLAG_CCF` : Computation Complete Flag
 - `CRYP_FLAG_RDERR` : Read Error Flag
 - `CRYP_FLAG_WRERR` : Write Error Flag

Return value:

- The new state of `__FLAG__` (TRUE or FALSE).

`__HAL_CRYP_CLEAR_FLAG`

Description:

- Clear the CRYP pending flag.

Parameters:

- `__HANDLE__`: specifies the CRYP

handle.

- __FLAG__: specifies the flag to clear.
This parameter can be one of the following values:
 - CRYP_CLEARFLAG_CCF : Computation Complete Clear Flag
 - CRYP_CLEARFLAG_RDERR : Read Error Clear
 - CRYP_CLEARFLAG_WRERR : Write Error Clear

Return value:

- None

_HAL_CRYP_ENABLE_IT

Description:

- Enable the CRYP interrupt.

Parameters:

- __HANDLE__: specifies the CRYP handle.
- __INTERRUPT__: CRYP Interrupt.

Return value:

- None

_HAL_CRYP_DISABLE_IT

Description:

- Disable the CRYP interrupt.

Parameters:

- __HANDLE__: specifies the CRYP handle.
- __INTERRUPT__: CRYP interrupt.

Return value:

- None

_HAL_CRYP_GET_IT_SOURCE

Description:

- Checks if the specified CRYP interrupt source is enabled or disabled.

Parameters:

- __HANDLE__: specifies the CRYP handle.
- __INTERRUPT__: CRYP interrupt source to check This parameter can be one of the following values:
 - CRYP_IT_CC : Computation Complete interrupt
 - CRYP_IT_ERR : Error interrupt (used for RDERR and WRERR)

Return value:

- State: of interruption (SET or RESET)

__HAL_CRYP_CLEAR_IT**Description:**

- Clear the CRYP pending IT.

Parameters:

- __HANDLE__: specifies the CRYP handle.
- __IT__: specifies the IT to clear. This parameter can be one of the following values:
 - CRYP_CLEARFLAG_CCF : Computation Complete Clear Flag
 - CRYP_CLEARFLAG_RDERR : Read Error Clear
 - CRYP_CLEARFLAG_WRERR : Write Error Clear

Return value:

- None

13 HAL CRYP Extension Driver

13.1 CRYPEx Firmware driver API description

13.1.1 Extended features functions

This section provides callback functions:

- Computation completed.

This section contains the following APIs:

- [*HAL_CRYPEx_ComputationCpltCallback\(\)*](#)

13.1.2 Detailed description of functions

HAL_CRYPEx_ComputationCpltCallback

Function name **void HAL_CRYPEx_ComputationCpltCallback
(CRYP_HandleTypeDef * hcryp)**

Function description Computation completed callbacks.

Parameters • **hcryp:** pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module

Return values • **None:**

14 HAL DAC Generic Driver

14.1 DAC Firmware driver registers structures

14.1.1 DAC_HandleTypeDef

Data Fields

- *DAC_TypeDef * Instance*
- *__IO HAL_DAC_StateTypeDef State*
- *HAL_LockTypeDef Lock*
- *DMA_HandleTypeDef * DMA_Handle1*
- *DMA_HandleTypeDef * DMA_Handle2*
- *__IO uint32_t ErrorCode*

Field Documentation

- ***DAC_TypeDef* DAC_HandleTypeDef::Instance***
Register base address
- ***__IO HAL_DAC_StateTypeDef DAC_HandleTypeDef::State***
DAC communication state
- ***HAL_LockTypeDef DAC_HandleTypeDef::Lock***
DAC locking object
- ***DMA_HandleTypeDef* DAC_HandleTypeDef::DMA_Handle1***
Pointer DMA handler for channel 1
- ***DMA_HandleTypeDef* DAC_HandleTypeDef::DMA_Handle2***
Pointer DMA handler for channel 2
- ***__IO uint32_t DAC_HandleTypeDef::ErrorCode***
DAC Error code

14.1.2 DAC_ChannelConfTypeDef

Data Fields

- *uint32_t DAC_Trigger*
- *uint32_t DAC_OutputBuffer*

Field Documentation

- ***uint32_t DAC_ChannelConfTypeDef::DAC_Trigger***
Specifies the external trigger for the selected DAC channel. This parameter can be a value of [DAC_trigger_selection](#)
- ***uint32_t DAC_ChannelConfTypeDef::DAC_OutputBuffer***
Specifies whether the DAC channel output buffer is enabled or disabled. This parameter can be a value of [DAC_output_buffer](#)

14.2 DAC Firmware driver API description

14.2.1 DAC Peripheral features

DAC Channels

The device integrates two 12-bit Digital Analog Converters that can be used independently or simultaneously (dual mode):

1. DAC channel1 with DAC_OUT1 (PA4) as output
2. DAC channel2 with DAC_OUT2 (PA5) as output

DAC Triggers

Digital to Analog conversion can be non-triggered using DAC_TRIGGER_NONE and DAC_OUT1/DAC_OUT2 is available once writing to DHRx register.

Digital to Analog conversion can be triggered by:

1. External event: EXTI Line 9 (any GPIOx_PIN_9) using DAC_TRIGGER_EXT_IT9. The used pin (GPIOx_PIN_9) must be configured in input mode.
2. Timers TRGO: TIM2, TIM4, TIM6, TIM7, TIM9 (DAC_Trigger_T2_TRGO, DAC_Trigger_T4_TRGO...)
3. Software using DAC_TRIGGER_SOFTWARE

DAC Buffer mode feature

Each DAC channel integrates an output buffer that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. To enable, the output buffer use sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;



Refer to the device datasheet for more details about output impedance value with and without output buffer.

DAC connect feature

Each DAC channel can be connected internally. To connect, use sConfig.DAC_ConnectOnChipPeripheral = DAC_CHIPCONNECT_ENABLE;

GPIO configurations guidelines

When a DAC channel is used (ex channel1 on PA4) and the other is not (ex channel1 on PA5 is configured in Analog and disabled). Channel1 may disturb channel2 as coupling effect. Note that there is no coupling on channel2 as soon as channel2 is turned on. Coupling on adjacent channel could be avoided as follows: when unused PA5 is configured as INPUT PULL-UP or DOWN. PA5 is configured in ANALOG just before it is turned on.

DAC wave generation feature

Both DAC channels can be used to generate

1. Noise wave using HAL_DACEx_NoiseWaveGenerate()
2. Triangle wave using HAL_DACEx_TriangleWaveGenerate()

DAC data format

The DAC data format can be:

1. 8-bit right alignment using DAC_ALIGN_8B_R
2. 12-bit left alignment using DAC_ALIGN_12B_L
3. 12-bit right alignment using DAC_ALIGN_12B_R

DAC data value to voltage correspondance

The analog output voltage on each DAC channel pin is determined by the following equation:

$$\text{DAC_OUTx} = \text{VREF+} * \text{DOR} / 4095$$

- with DOR is the Data Output Register

VREF+ is the input voltage reference (refer to the device datasheet)

e.g. To set DAC_OUT1 to 0.7V, use

- Assuming that VREF+ = 3.3V, DAC_OUT1 = $(3.3 * 868) / 4095 = 0.7V$

DMA requests

A DMA1 request can be generated when an external trigger (but not a software trigger) occurs if DMA1 requests are enabled using HAL_DAC_Start_DMA()

DMA1 requests are mapped as following:

- DAC channel1 : mapped on DMA1 channel2 which must be already configured
- DAC channel2 : mapped on DMA1 channel3 which must be already configured For Dual mode and specific signal (Triangle and noise) generation please refer to Extension Features Driver description

14.2.2 How to use this driver

- DAC APB clock must be enabled to get write access to DAC registers using HAL_DAC_Init()
- Configure DAC_OUTx (DAC_OUT1: PA4, DAC_OUT2: PA5) in analog mode.
- Configure the DAC channel using HAL_DAC_ConfigChannel() function.
- Enable the DAC channel using HAL_DAC_Start() or HAL_DAC_Start_DMA functions

Polling mode IO operation

- Start the DAC peripheral using HAL_DAC_Start()
- To read the DAC last data output value, use the HAL_DAC_GetValue() function.
- Stop the DAC peripheral using HAL_DAC_Stop()

DMA mode IO operation

- Start the DAC peripheral using HAL_DAC_Start_DMA(), at this stage the user specify the length of data to be transferred at each end of conversion
- At the middle of data transfer HAL_DACEx_ConvHalfCpltCallbackCh1() or HAL_DACEx_ConvHalfCpltCallbackCh2() function is executed and user can add his own code by customization of function pointer HAL_DAC_ConvHalfCpltCallbackCh1 or HAL_DAC_ConvHalfCpltCallbackCh2
- At The end of data transfer HAL_DAC_ConvCpltCallbackCh1() or HAL_DAC_ConvCpltCallbackCh2() function is executed and user can add his own code by customization of function pointer HAL_DAC_ConvCpltCallbackCh1 or HAL_DAC_ConvCpltCallbackCh2
- In case of transfer Error, HAL_DAC_ErrorCallbackCh1() or HAL_DACEx_ErrorCallbackCh2() function is executed and user can add his own code by customization of function pointer HAL_DAC_ErrorCallbackCh1 or HAL_DACEx_ErrorCallbackCh2
- For STM32F100x devices with specific feature: DMA underrun. In case of DMA underrun, DAC interruption triggers and execute internal function HAL_DAC_IRQHandler. HAL_DAC_DMAUnderrunCallbackCh1() or



HAL_DACEx_DMAUnderrunCallbackCh2() function is executed and user can add his own code by customization of function pointer HAL_DAC_DMAUnderrunCallbackCh1 or HAL_DACEx_DMAUnderrunCallbackCh2 add his own code by customization of function pointer HAL_DAC_ErrorCallbackCh1

- Stop the DAC peripheral using HAL_DAC_Stop_DMA()

DAC HAL driver macros list

Below the list of most used macros in DAC HAL driver.

- __HAL_DAC_ENABLE : Enable the DAC peripheral
- __HAL_DAC_DISABLE : Disable the DAC peripheral
- __HAL_DAC_CLEAR_FLAG: Clear the DAC's pending flags
- __HAL_DAC_GET_FLAG: Get the selected DAC's flag status



You can refer to the DAC HAL driver header file for more useful macros

14.2.3 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize and configure the DAC.
- De-initialize the DAC.

This section contains the following APIs:

- [*HAL_DAC_Init\(\)*](#)
- [*HAL_DAC_DelInit\(\)*](#)
- [*HAL_DAC_MspInit\(\)*](#)
- [*HAL_DAC_MspDelInit\(\)*](#)

14.2.4 IO operation functions

This section provides functions allowing to:

- Start conversion.
- Stop conversion.
- Start conversion and enable DMA transfer.
- Stop conversion and disable DMA transfer.
- Get result of conversion.

This section contains the following APIs:

- [*HAL_DAC_Start\(\)*](#)
- [*HAL_DAC_Stop\(\)*](#)
- [*HAL_DAC_Start_DMA\(\)*](#)
- [*HAL_DAC_Stop_DMA\(\)*](#)
- [*HAL_DAC_GetValue\(\)*](#)
- [*HAL_DAC_IRQHandler\(\)*](#)
- [*HAL_DAC_ConvCpltCallbackCh1\(\)*](#)
- [*HAL_DAC_ConvHalfCpltCallbackCh1\(\)*](#)
- [*HAL_DAC_ErrorCallbackCh1\(\)*](#)
- [*HAL_DAC_DMAUnderrunCallbackCh1\(\)*](#)
- [*HAL_DAC_SetValue\(\)*](#)
- [*HAL_DAC_ConfigChannel\(\)*](#)

- [*HAL_DAC_GetState\(\)*](#)
- [*HAL_DAC_GetError\(\)*](#)

14.2.5 Peripheral Control functions

This section provides functions allowing to:

- Configure channels.
- Set the specified data holding register value for DAC channel.

This section contains the following APIs:

- [*HAL_DAC_ConfigChannel\(\)*](#)
- [*HAL_DAC_SetValue\(\)*](#)

14.2.6 Peripheral State and Errors functions

This subsection provides functions allowing to

- Check the DAC state.
- Check the DAC Errors.

This section contains the following APIs:

- [*HAL_DAC_GetState\(\)*](#)
- [*HAL_DAC_GetError\(\)*](#)

14.2.7 Detailed description of functions

HAL_DAC_Init

Function name	HAL_StatusTypeDef HAL_DAC_Init (DAC_HandleTypeDef * hdac)
Function description	Initializes the DAC peripheral according to the specified parameters in the DAC_InitStruct.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DAC_DeInit

Function name	HAL_StatusTypeDef HAL_DAC_DeInit (DAC_HandleTypeDef * hdac)
Function description	Deinitializes the DAC peripheral registers to their default reset values.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DAC_MspInit

Function name	void HAL_DAC_MspInit (DAC_HandleTypeDef * hdac)
Function description	Initializes the DAC MSP.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that

contains the configuration information for the specified DAC.

Return values

- **None:**

HAL_DAC_MspDelInit

Function name **void HAL_DAC_MspDelInit (DAC_HandleTypeDef * hdac)**

Function description Delinitializes the DAC MSP.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.

Return values

- **None:**

HAL_DAC_Start

Function name **HAL_StatusTypeDef HAL_DAC_Start (DAC_HandleTypeDef * hdac, uint32_t Channel)**

Function description Enables DAC and starts conversion of channel.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
- **Channel:** The selected DAC channel. This parameter can be one of the following values:
 - DAC_CHANNEL_1: DAC Channel1 selected
 - DAC_CHANNEL_2: DAC Channel2 selected

Return values

- **HAL:** status

HAL_DAC_Stop

Function name **HAL_StatusTypeDef HAL_DAC_Stop (DAC_HandleTypeDef * hdac, uint32_t Channel)**

Function description Disables DAC and stop conversion of channel.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
- **Channel:** The selected DAC channel. This parameter can be one of the following values:
 - DAC_CHANNEL_1: DAC Channel1 selected
 - DAC_CHANNEL_2: DAC Channel2 selected

Return values

- **HAL:** status

HAL_DAC_Start_DMA

Function name **HAL_StatusTypeDef HAL_DAC_Start_DMA (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t * pData, uint32_t Length, uint32_t Alignment)**

Function description Enables DAC and starts conversion of channel.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
- **Channel:** The selected DAC channel. This parameter can be one of the following values:
 - DAC_CHANNEL_1: DAC Channel1 selected

- DAC_CHANNEL_2: DAC Channel2 selected
 - **pData:** The destination peripheral Buffer address.
 - **Length:** The length of data to be transferred from memory to DAC peripheral
 - **Alignment:** Specifies the data alignment for DAC channel. This parameter can be one of the following values:
 - DAC_ALIGN_8B_R: 8bit right data alignment selected
 - DAC_ALIGN_12B_L: 12bit left data alignment selected
 - DAC_ALIGN_12B_R: 12bit right data alignment selected
- Return values**
- **HAL:** status

HAL_DAC_Stop_DMA

Function name	HAL_StatusTypeDef HAL_DAC_Stop_DMA (DAC_HandleTypeDef * hdac, uint32_t Channel)
Function description	Disables DAC and stop conversion of channel.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. • Channel: The selected DAC channel. This parameter can be one of the following values: <ul style="list-style-type: none"> – DAC_CHANNEL_1: DAC Channel1 selected – DAC_CHANNEL_2: DAC Channel2 selected
Return values	• HAL: status

HAL_DAC_SetValue

Function name	HAL_StatusTypeDef HAL_DAC_SetValue (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data)
Function description	Set the specified data holding register value for DAC channel.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. • Channel: The selected DAC channel. This parameter can be one of the following values: <ul style="list-style-type: none"> – DAC_CHANNEL_1: DAC Channel1 selected – DAC_CHANNEL_2: DAC Channel2 selected • Alignment: Specifies the data alignment. This parameter can be one of the following values: <ul style="list-style-type: none"> – DAC_ALIGN_8B_R: 8bit right data alignment selected – DAC_ALIGN_12B_L: 12bit left data alignment selected – DAC_ALIGN_12B_R: 12bit right data alignment selected • Data: Data to be loaded in the selected data holding register.
Return values	• HAL: status

HAL_DAC_GetValue

Function name	uint32_t HAL_DAC_GetValue (DAC_HandleTypeDef * hdac, uint32_t Channel)
Function description	Returns the last data output value of the selected DAC channel.

Parameters	<ul style="list-style-type: none"> hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. Channel: The selected DAC channel. This parameter can be one of the following values: <ul style="list-style-type: none"> – DAC_CHANNEL_1: DAC Channel1 selected – DAC_CHANNEL_2: DAC Channel2 selected
Return values	<ul style="list-style-type: none"> The: selected DAC channel data output value.

HAL_DAC_ConfigChannel

Function name	HAL_StatusTypeDef HAL_DAC_ConfigChannel (DAC_HandleTypeDef * hdac, DAC_ChannelConfTypeDef * sConfig, uint32_t Channel)
Function description	Configures the selected DAC channel.
Parameters	<ul style="list-style-type: none"> hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. sConfig: DAC configuration structure. Channel: The selected DAC channel. This parameter can be one of the following values: <ul style="list-style-type: none"> – DAC_CHANNEL_1: DAC Channel1 selected – DAC_CHANNEL_2: DAC Channel2 selected
Return values	<ul style="list-style-type: none"> HAL: status

HAL_DAC_GetState

Function name	HAL_DAC_StateTypeDef HAL_DAC_GetState (DAC_HandleTypeDef * hdac)
Function description	return the DAC state
Parameters	<ul style="list-style-type: none"> hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> HAL: state

HAL_DAC_IRQHandler

Function name	void HAL_DAC_IRQHandler (DAC_HandleTypeDef * hdac)
Function description	Handles DAC interrupt request.
Parameters	<ul style="list-style-type: none"> hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> None:

HAL_DAC_GetError

Function name	uint32_t HAL_DAC_GetError (DAC_HandleTypeDef * hdac)
Function description	Return the DAC error code.
Parameters	<ul style="list-style-type: none"> hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> DAC: Error Code

HAL_DAC_ConvCpltCallbackCh1

Function name	void HAL_DAC_ConvCpltCallbackCh1 (DAC_HandleTypeDef * hdac)
Function description	Conversion complete callback in non blocking mode for Channel1.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> • None:

HAL_DAC_ConvHalfCpltCallbackCh1

Function name	void HAL_DAC_ConvHalfCpltCallbackCh1 (DAC_HandleTypeDef * hdac)
Function description	Conversion half DMA transfer callback in non blocking mode for Channel1.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> • None:

HAL_DAC_ErrorCallbackCh1

Function name	void HAL_DAC_ErrorCallbackCh1 (DAC_HandleTypeDef * hdac)
Function description	Error DAC callback for Channel1.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> • None:

HAL_DAC_DMAUnderrunCallbackCh1

Function name	void HAL_DAC_DMAUnderrunCallbackCh1 (DAC_HandleTypeDef * hdac)
Function description	DMA underrun DAC callback for channel1.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
Return values	<ul style="list-style-type: none"> • None:

14.3 DAC Firmware driver defines

14.3.1 DAC

DAC Channel selection

DAC_CHANNEL_1

DAC_CHANNEL_2

DAC data alignment

DAC_ALIGN_12B_R

DAC_ALIGN_12B_L

DAC_ALIGN_8B_R

DAC Error Code

HAL_DAC_ERROR_NONE No error

HAL_DAC_ERROR_DMAUNDERRUNCH1 DAC channel1 DMA underrun error

HAL_DAC_ERROR_DMAUNDERRUNCH2 DAC channel2 DMA underrun error

HAL_DAC_ERROR_DMA DMA error

DAC Exported Macros

__HAL_DAC_RESET_HANDLE_STATE **Description:**

- Reset DAC handle state.

Parameters:

- __HANDLE__: specifies the DAC handle.

Return value:

- None

__HAL_DAC_ENABLE

Description:

- Enable the DAC channel.

Parameters:

- __HANDLE__: specifies the DAC handle.
- __DAC_Channel__: specifies the DAC channel

Return value:

- None

__HAL_DAC_DISABLE

Description:

- Disable the DAC channel.

Parameters:

- __HANDLE__: specifies the DAC handle
- __DAC_Channel__: specifies the DAC channel.

Return value:

- None

__HAL_DAC_ENABLE_IT

Description:

- Enable the DAC interrupt.

Parameters:

- __HANDLE__: specifies the DAC handle
- __INTERRUPT__: specifies the DAC interrupt. This parameter can be any combination of the following values:
 - DAC_IT_DMAUDR1: DAC channel 1

- DMA underrun interrupt
- DAC_IT_DMAUDR2: DAC channel 2 DMA underrun interrupt

Return value:

- None

__HAL_DAC_DISABLE_IT

- Disable the DAC interrupt.

Parameters:

- __HANDLE__: specifies the DAC handle
- __INTERRUPT__: specifies the DAC interrupt. This parameter can be any combination of the following values:
 - DAC_IT_DMAUDR1: DAC channel 1 DMA underrun interrupt
 - DAC_IT_DMAUDR2: DAC channel 2 DMA underrun interrupt

Return value:

- None

__HAL_DAC_GET_IT_SOURCE

- Checks if the specified DAC interrupt source is enabled or disabled.

Parameters:

- __HANDLE__: DAC handle
- __INTERRUPT__: DAC interrupt source to check. This parameter can be any combination of the following values:
 - DAC_IT_DMAUDR1: DAC channel 1 DMA underrun interrupt
 - DAC_IT_DMAUDR2: DAC channel 2 DMA underrun interrupt

Return value:

- State: of interruption (SET or RESET)

__HAL_DAC_GET_FLAG

- Get the selected DAC's flag status.

Parameters:

- __HANDLE__: specifies the DAC handle.
- __FLAG__: specifies the DAC flag to get. This parameter can be any combination of the following values:
 - DAC_FLAG_DMAUDR1: DAC channel 1 DMA underrun flag
 - DAC_FLAG_DMAUDR2: DAC channel 2 DMA underrun flag

Return value:

- None

_HAL_DAC_CLEAR_FLAG**Description:**

- Clear the DAC's flag.

Parameters:

- HANDLE: specifies the DAC handle.
- FLAG: specifies the DAC flag to clear. This parameter can be any combination of the following values:
 - DAC_FLAG_DMAUDR1: DAC channel 1 DMA underrun flag
 - DAC_FLAG_DMAUDR2: DAC channel 2 DMA underrun flag

Return value:

- None

DAC flags definition

DAC_FLAG_DMAUDR1

DAC_FLAG_DMAUDR2

DAC IT definition

DAC_IT_DMAUDR1

DAC_IT_DMAUDR2

DAC output buffer

DAC_OUTPUTBUFFER_ENABLE

DAC_OUTPUTBUFFER_DISABLE

DAC trigger selection

DAC_TRIGGER_NONE	Conversion is automatic once the DAC1_DHRxxxx register has been loaded, and not by external trigger
------------------	---

DAC_TRIGGER_T6_TRGO	TIM6 TRGO selected as external conversion trigger for DAC channel
---------------------	---

DAC_TRIGGER_T7_TRGO	TIM7 TRGO selected as external conversion trigger for DAC channel
---------------------	---

DAC_TRIGGER_T9_TRGO	TIM9 TRGO selected as external conversion trigger for DAC channel
---------------------	---

DAC_TRIGGER_T2_TRGO	TIM2 TRGO selected as external conversion trigger for DAC channel
---------------------	---

DAC_TRIGGER_T4_TRGO	TIM4 TRGO selected as external conversion trigger for DAC channel
---------------------	---

DAC_TRIGGER_EXT_IT9	EXTI Line9 event selected as external conversion trigger for DAC channel
---------------------	--

DAC_TRIGGER_SOFTWARE	Conversion started by software trigger for DAC channel
----------------------	--

15 HAL DAC Extension Driver

15.1 DACEx Firmware driver API description

15.1.1 How to use this driver

- When Dual mode is enabled (i.e DAC Channel1 and Channel2 are used simultaneously) : Use HAL_DACEx_DualGetValue() to get digital data to be converted and use HAL_DACEx_DualSetValue() to set digital value to converted simultaneously in Channel 1 and Channel 2.
- Use HAL_DACEx_TriangleWaveGenerate() to generate Triangle signal.
- Use HAL_DACEx_NoiseWaveGenerate() to generate Noise signal.

15.1.2 Extended features functions

This section provides functions allowing to:

- Start conversion.
- Stop conversion.
- Start conversion and enable DMA transfer.
- Stop conversion and disable DMA transfer.
- Get result of conversion.
- Get result of dual mode conversion.

This section contains the following APIs:

- [*HAL_DACEx_DualGetValue\(\)*](#)
- [*HAL_DACEx_TriangleWaveGenerate\(\)*](#)
- [*HAL_DACEx_NoiseWaveGenerate\(\)*](#)
- [*HAL_DACEx_DualSetValue\(\)*](#)
- [*HAL_DACEx_ConvCpltCallbackCh2\(\)*](#)
- [*HAL_DACEx_ConvHalfCpltCallbackCh2\(\)*](#)
- [*HAL_DACEx_ErrorCallbackCh2\(\)*](#)
- [*HAL_DACEx_DMAUnderrunCallbackCh2\(\)*](#)

15.1.3 Detailed description of functions

HAL_DACEx_DualGetValue

Function name **uint32_t HAL_DACEx_DualGetValue (DAC_HandleTypeDef * hdac)**

Function description Returns the last data output value of the selected DAC channel.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.

Return values

- **The:** selected DAC channel data output value.

HAL_DACEx_TriangleWaveGenerate

Function name **HAL_StatusTypeDef HAL_DACEx_TriangleWaveGenerate (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t Amplitude)**

Function description	Enables or disables the selected DAC channel wave generation.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. • Channel: The selected DAC channel. This parameter can be one of the following values: DAC_CHANNEL_1 / DAC_CHANNEL_2 • Amplitude: Select max triangle amplitude. This parameter can be one of the following values: <ul style="list-style-type: none"> – DAC_TRIANGLEAMPLITUDE_1: Select max triangle amplitude of 1 – DAC_TRIANGLEAMPLITUDE_3: Select max triangle amplitude of 3 – DAC_TRIANGLEAMPLITUDE_7: Select max triangle amplitude of 7 – DAC_TRIANGLEAMPLITUDE_15: Select max triangle amplitude of 15 – DAC_TRIANGLEAMPLITUDE_31: Select max triangle amplitude of 31 – DAC_TRIANGLEAMPLITUDE_63: Select max triangle amplitude of 63 – DAC_TRIANGLEAMPLITUDE_127: Select max triangle amplitude of 127 – DAC_TRIANGLEAMPLITUDE_255: Select max triangle amplitude of 255 – DAC_TRIANGLEAMPLITUDE_511: Select max triangle amplitude of 511 – DAC_TRIANGLEAMPLITUDE_1023: Select max triangle amplitude of 1023 – DAC_TRIANGLEAMPLITUDE_2047: Select max triangle amplitude of 2047 – DAC_TRIANGLEAMPLITUDE_4095: Select max triangle amplitude of 4095
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DACEx_NoiseWaveGenerate

Function name	HAL_StatusTypeDef HAL_DACEx_NoiseWaveGenerate(DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t Amplitude)
Function description	Enables or disables the selected DAC channel wave generation.
Parameters	<ul style="list-style-type: none"> • hdac: pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. • Channel: The selected DAC channel. This parameter can be one of the following values: DAC_CHANNEL_1 / DAC_CHANNEL_2 • Amplitude: Unmask DAC channel LFSR for noise wave generation. This parameter can be one of the following values: <ul style="list-style-type: none"> – DAC_LFSRUNMASK_BIT0: Unmask DAC channel LFSR bit0 for noise wave generation – DAC_LFSRUNMASK_BITS1_0: Unmask DAC channel LFSR bit[1:0] for noise wave generation

- DAC_LFSRUNMASK_BITS2_0: Unmask DAC channel LFSR bit[2:0] for noise wave generation
- DAC_LFSRUNMASK_BITS3_0: Unmask DAC channel LFSR bit[3:0] for noise wave generation
- DAC_LFSRUNMASK_BITS4_0: Unmask DAC channel LFSR bit[4:0] for noise wave generation
- DAC_LFSRUNMASK_BITS5_0: Unmask DAC channel LFSR bit[5:0] for noise wave generation
- DAC_LFSRUNMASK_BITS6_0: Unmask DAC channel LFSR bit[6:0] for noise wave generation
- DAC_LFSRUNMASK_BITS7_0: Unmask DAC channel LFSR bit[7:0] for noise wave generation
- DAC_LFSRUNMASK_BITS8_0: Unmask DAC channel LFSR bit[8:0] for noise wave generation
- DAC_LFSRUNMASK_BITS9_0: Unmask DAC channel LFSR bit[9:0] for noise wave generation
- DAC_LFSRUNMASK_BITS10_0: Unmask DAC channel LFSR bit[10:0] for noise wave generation
- DAC_LFSRUNMASK_BITS11_0: Unmask DAC channel LFSR bit[11:0] for noise wave generation

Return values

- **HAL:** status

HAL_DACEx_DualSetValue

Function name

**HAL_StatusTypeDef HAL_DACEx_DualSetValue
(DAC_HandleTypeDef * hdac, uint32_t Alignment, uint32_t Data1, uint32_t Data2)**

Function description

Set the specified data holding register value for dual DAC channel.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
- **Alignment:** Specifies the data alignment for dual channel DAC. This parameter can be one of the following values:
DAC_ALIGN_8B_R: 8bit right data alignment selected
DAC_ALIGN_12B_L: 12bit left data alignment selected
DAC_ALIGN_12B_R: 12bit right data alignment selected
- **Data1:** Data for DAC Channel2 to be loaded in the selected data holding register.
- **Data2:** Data for DAC Channel1 to be loaded in the selected data holding register.

Return values

- **HAL:** status

Notes

- In dual mode, a unique register access is required to write in both DAC channels at the same time.

HAL_DACEx_ConvCpltCallbackCh2

Function name

**void HAL_DACEx_ConvCpltCallbackCh2
(DAC_HandleTypeDef * hdac)**

Function description

Conversion complete callback in non blocking mode for Channel2.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.

Return values

- **None:**

HAL_DACEx_ConvHalfCpltCallbackCh2**Function name**

**void HAL_DACEx_ConvHalfCpltCallbackCh2
(DAC_HandleTypeDef * hdac)**

Function description

Conversion half DMA transfer callback in non blocking mode for Channel2.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.

Return values

- **None:**

HAL_DACEx_ErrorCallbackCh2**Function name**

void HAL_DACEx_ErrorCallbackCh2 (DAC_HandleTypeDef * hdac)

Function description

Error DAC callback for Channel2.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.

Return values

- **None:**

HAL_DACEx_DMAUnderrunCallbackCh2**Function name**

**void HAL_DACEx_DMAUnderrunCallbackCh2
(DAC_HandleTypeDef * hdac)**

Function description

DMA underrun DAC callback for channel2.

Parameters

- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.

Return values

- **None:**

DAC_DMAConvCpltCh2**Function name**

void DAC_DMAConvCpltCh2 (DMA_HandleTypeDef * hdma)

Function description

DMA conversion complete callback.

Parameters

- **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module.

Return values

- **None:**

DAC_DMAMhalfConvCpltCh2**Function name**

void DAC_DMAMhalfConvCpltCh2 (DMA_HandleTypeDef * hdma)

Function description

DMA half transfer complete callback.

Parameters

- **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA

module.

Return values

- **None:**

DAC_DMAErrorCh2

Function name **void DAC_DMAErrorCh2 (DMA_HandleTypeDef * hdma)**

Function description DMA error callback.

Parameters

- **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module.

Return values

- **None:**

15.2 DACEx Firmware driver defines

15.2.1 DACEx

DACEx Ifsrunmask triangleamplitude

DAC_LFSRUNMASK_BIT0	Unmask DAC channel LFSR bit0 for noise wave generation
DAC_LFSRUNMASK_BITS1_0	Unmask DAC channel LFSR bit[1:0] for noise wave generation
DAC_LFSRUNMASK_BITS2_0	Unmask DAC channel LFSR bit[2:0] for noise wave generation
DAC_LFSRUNMASK_BITS3_0	Unmask DAC channel LFSR bit[3:0] for noise wave generation
DAC_LFSRUNMASK_BITS4_0	Unmask DAC channel LFSR bit[4:0] for noise wave generation
DAC_LFSRUNMASK_BITS5_0	Unmask DAC channel LFSR bit[5:0] for noise wave generation
DAC_LFSRUNMASK_BITS6_0	Unmask DAC channel LFSR bit[6:0] for noise wave generation
DAC_LFSRUNMASK_BITS7_0	Unmask DAC channel LFSR bit[7:0] for noise wave generation
DAC_LFSRUNMASK_BITS8_0	Unmask DAC channel LFSR bit[8:0] for noise wave generation
DAC_LFSRUNMASK_BITS9_0	Unmask DAC channel LFSR bit[9:0] for noise wave generation
DAC_LFSRUNMASK_BITS10_0	Unmask DAC channel LFSR bit[10:0] for noise wave generation
DAC_LFSRUNMASK_BITS11_0	Unmask DAC channel LFSR bit[11:0] for noise wave generation
DAC_TRIANGLEAMPLITUDE_1	Select max triangle amplitude of 1
DAC_TRIANGLEAMPLITUDE_3	Select max triangle amplitude of 3
DAC_TRIANGLEAMPLITUDE_7	Select max triangle amplitude of 7

DAC_TRIANGLEAMPLITUDE_15	Select max triangle amplitude of 15
DAC_TRIANGLEAMPLITUDE_31	Select max triangle amplitude of 31
DAC_TRIANGLEAMPLITUDE_63	Select max triangle amplitude of 63
DAC_TRIANGLEAMPLITUDE_127	Select max triangle amplitude of 127
DAC_TRIANGLEAMPLITUDE_255	Select max triangle amplitude of 255
DAC_TRIANGLEAMPLITUDE_511	Select max triangle amplitude of 511
DAC_TRIANGLEAMPLITUDE_1023	Select max triangle amplitude of 1023
DAC_TRIANGLEAMPLITUDE_2047	Select max triangle amplitude of 2047
DAC_TRIANGLEAMPLITUDE_4095	Select max triangle amplitude of 4095
IS_DAC_LFSR_UNMASK_TRIANGLE_AMPLITUDE	
DACEx wave generation	
DAC_WAVE_NOISE	
DAC_WAVE_TRIANGLE	

16 HAL DMA Generic Driver

16.1 DMA Firmware driver registers structures

16.1.1 DMA_InitTypeDef

Data Fields

- *uint32_t Direction*
- *uint32_t PeriphInc*
- *uint32_t MemInc*
- *uint32_t PeriphDataAlignment*
- *uint32_t MemDataAlignment*
- *uint32_t Mode*
- *uint32_t Priority*

Field Documentation

- ***uint32_t DMA_InitTypeDef::Direction***
Specifies if the data will be transferred from memory to peripheral, from memory to memory or from peripheral to memory. This parameter can be a value of [**DMA_Data_transfer_direction**](#)
- ***uint32_t DMA_InitTypeDef::PeriphInc***
Specifies whether the Peripheral address register should be incremented or not. This parameter can be a value of [**DMA_Peripheral_incremented_mode**](#)
- ***uint32_t DMA_InitTypeDef::MemInc***
Specifies whether the memory address register should be incremented or not. This parameter can be a value of [**DMA_Memory_incremented_mode**](#)
- ***uint32_t DMA_InitTypeDef::PeriphDataAlignment***
Specifies the Peripheral data width. This parameter can be a value of [**DMA_Peripheral_data_size**](#)
- ***uint32_t DMA_InitTypeDef::MemDataAlignment***
Specifies the Memory data width. This parameter can be a value of [**DMA_Memory_data_size**](#)
- ***uint32_t DMA_InitTypeDef::Mode***
Specifies the operation mode of the DMAy Channelx. This parameter can be a value of [**DMA_mode**](#)
Note:The circular buffer mode cannot be used if the memory-to-memory data transfer is configured on the selected Channel
- ***uint32_t DMA_InitTypeDef::Priority***
Specifies the software priority for the DMAy Channelx. This parameter can be a value of [**DMA_Priority_level**](#)

16.1.2 __DMA_HandleTypeDef

Data Fields

- *DMA_Channel_TypeDef * Instance*
- *DMA_InitTypeDef Init*
- *HAL_LockTypeDef Lock*
- *__IO HAL_DMA_StateTypeDef State*
- *void * Parent*
- *void(* XferCpltCallback*
- *void(* XferHalfCpltCallback*

- `void(* XferErrorCallback`
- `void(* XferAbortCallback`
- `__IO uint32_t ErrorCode`
- `DMA_TypeDef * DmaBaseAddress`
- `uint32_t ChannelIndex`

Field Documentation

- `DMA_Channel_TypeDef* __DMA_HandleTypeDef::Instance`
Register base address
- `DMA_InitTypeDef __DMA_HandleTypeDef::Init`
DMA communication parameters
- `HAL_LockTypeDef __DMA_HandleTypeDef::Lock`
DMA locking object
- `__IO HAL_DMA_StateTypeDef __DMA_HandleTypeDef::State`
DMA transfer state
- `void* __DMA_HandleTypeDef::Parent`
Parent object state
- `void(* __DMA_HandleTypeDef::XferCpltCallback)(struct __DMA_HandleTypeDef *hdma)`
DMA transfer complete callback
- `void(* __DMA_HandleTypeDef::XferHalfCpltCallback)(struct __DMA_HandleTypeDef *hdma)`
DMA Half transfer complete callback
- `void(* __DMA_HandleTypeDef::XferErrorCallback)(struct __DMA_HandleTypeDef *hdma)`
DMA transfer error callback
- `void(* __DMA_HandleTypeDef::XferAbortCallback)(struct __DMA_HandleTypeDef *hdma)`
DMA transfer abort callback
- `__IO uint32_t __DMA_HandleTypeDef::ErrorCode`
DMA Error code
- `DMA_TypeDef* __DMA_HandleTypeDef::DmaBaseAddress`
DMA Channel Base Address
- `uint32_t __DMA_HandleTypeDef::ChannelIndex`
DMA Channel Index

16.2 DMA Firmware driver API description

16.2.1 How to use this driver

1. Enable and configure the peripheral to be connected to the DMA Channel (except for internal SRAM / FLASH memories: no initialization is necessary). Please refer to the Reference manual for connection between peripherals and DMA requests.
2. For a given Channel, program the required configuration through the following parameters: Channel request, Transfer Direction, Source and Destination data formats, Circular or Normal mode, Channel Priority level, Source and Destination Increment mode using HAL_DMA_Init() function.
3. Use HAL_DMA_GetState() function to return the DMA state and HAL_DMA_GetError() in case of error detection.
4. Use HAL_DMA_Abort() function to abort the current transfer In Memory-to-Memory transfer mode, Circular mode is not allowed.

Polling mode IO operation

- Use HAL_DMA_Start() to start DMA transfer after the configuration of Source address and destination address and the Length of data to be transferred
- Use HAL_DMA_PollForTransfer() to poll for the end of current transfer, in this case a fixed Timeout can be configured by User depending from his application.

Interrupt mode IO operation

- Configure the DMA interrupt priority using HAL_NVIC_SetPriority()
- Enable the DMA IRQ handler using HAL_NVIC_EnableIRQ()
- Use HAL_DMA_Start_IT() to start DMA transfer after the configuration of Source address and destination address and the Length of data to be transferred. In this case the DMA interrupt is configured
- Use HAL_DMA_IRQHandler() called under DMA_IRQHandler() Interrupt subroutine
- At the end of data transfer HAL_DMA_IRQHandler() function is executed and user can add his own function by customization of function pointer XferCpltCallback and XferErrorCallback (i.e. a member of DMA handle structure).

DMA HAL driver macros list

Below the list of most used macros in DMA HAL driver.

- __HAL_DMA_ENABLE: Enable the specified DMA Channel.
- __HAL_DMA_DISABLE: Disable the specified DMA Channel.
- __HAL_DMA_GET_FLAG: Get the DMA Channel pending flags.
- __HAL_DMA_CLEAR_FLAG: Clear the DMA Channel pending flags.
- __HAL_DMA_ENABLE_IT: Enable the specified DMA Channel interrupts.
- __HAL_DMA_DISABLE_IT: Disable the specified DMA Channel interrupts.
- __HAL_DMA_GET_IT_SOURCE: Check whether the specified DMA Channel interrupt has occurred or not.



You can refer to the DMA HAL driver header file for more useful macros

16.2.2 Initialization and de-initialization functions

This section provides functions allowing to initialize the DMA Channel source and destination addresses, incrementation and data sizes, transfer direction, circular/normal mode selection, memory-to-memory mode selection and Channel priority value.

The HAL_DMA_Init() function follows the DMA configuration procedures as described in reference manual.

This section contains the following APIs:

- [*HAL_DMA_Init\(\)*](#)
- [*HAL_DMA_DelInit\(\)*](#)

16.2.3 IO operation functions

This section provides functions allowing to:

- Configure the source, destination address and data length and Start DMA transfer
- Configure the source, destination address and data length and Start DMA transfer with interrupt

- Abort DMA transfer
- Poll for transfer complete
- Handle DMA interrupt request

This section contains the following APIs:

- [*HAL_DMA_Start\(\)*](#)
- [*HAL_DMA_Start_IT\(\)*](#)
- [*HAL_DMA_Abort\(\)*](#)
- [*HAL_DMA_Abort_IT\(\)*](#)
- [*HAL_DMA_PollForTransfer\(\)*](#)
- [*HAL_DMA_IRQHandler\(\)*](#)
- [*HAL_DMA_RegisterCallback\(\)*](#)
- [*HAL_DMA_UnRegisterCallback\(\)*](#)

16.2.4 Peripheral State and Errors functions

This subsection provides functions allowing to

- Check the DMA state
- Get error code

This section contains the following APIs:

- [*HAL_DMA_GetState\(\)*](#)
- [*HAL_DMA_GetError\(\)*](#)

16.2.5 Detailed description of functions

HAL_DMA_Init

Function name	HAL_StatusTypeDef HAL_DMA_Init (DMA_HandleTypeDef * hdma)
Function description	Initialize the DMA according to the specified parameters in the DMA_InitTypeDef and initialize the associated handle.
Parameters	<ul style="list-style-type: none"> • hdma: Pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DMA_DeInit

Function name	HAL_StatusTypeDef HAL_DMA_DeInit (DMA_HandleTypeDef * hdma)
Function description	Deinitialize the DMA peripheral.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DMA_Start

Function name	HAL_StatusTypeDef HAL_DMA_Start (DMA_HandleTypeDef * hdma, uint32_t SrcAddress, uint32_t DstAddress, uint32_t
---------------	--

DataLength)

Function description	Start the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • hdma: : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel. • SrcAddress: The source memory Buffer address • DstAddress: The destination memory Buffer address • DataLength: The length of data to be transferred from source to destination
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DMA_Start_IT

Function name	HAL_StatusTypeDef HAL_DMA_Start_IT (DMA_HandleTypeDef * hdma, uint32_t SrcAddress, uint32_t DstAddress, uint32_t DataLength)
Function description	Start the DMA Transfer with interrupt enabled.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel. • SrcAddress: The source memory Buffer address • DstAddress: The destination memory Buffer address • DataLength: The length of data to be transferred from source to destination
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DMA_Abort

Function name	HAL_StatusTypeDef HAL_DMA_Abort (DMA_HandleTypeDef * hdma)
Function description	Abort the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • hdma: : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DMA_Abort_IT

Function name	HAL_StatusTypeDef HAL_DMA_Abort_IT (DMA_HandleTypeDef * hdma)
Function description	Aborts the DMA Transfer in Interrupt mode.
Parameters	<ul style="list-style-type: none"> • hdma: : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Stream.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DMA_PollForTransfer

Function name	HAL_StatusTypeDef HAL_DMA_PollForTransfer (DMA_HandleTypeDef * hdma, uint32_t CompleteLevel, uint32_t Timeout)
Function description	Polling for transfer complete.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel. • CompleteLevel: Specifies the DMA level complete. • Timeout: Timeout duration.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DMA_IRQHandler

Function name	void HAL_DMA_IRQHandler (DMA_HandleTypeDef * hdma)
Function description	Handle DMA interrupt request.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
Return values	<ul style="list-style-type: none"> • None:

HAL_DMA_RegisterCallback

Function name	HAL_StatusTypeDef HAL_DMA_RegisterCallback (DMA_HandleTypeDef * hdma, HAL_DMA_CallbackIDTypeDef CallbackID, void(*)(DMA_HandleTypeDef * _hdma) pCallback)
Function description	Register callbacks.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Stream. • CallbackID: User Callback identifier a HAL_DMA_CallbackIDTypeDef ENUM as parameter. • pCallback: pointer to private callback function which has pointer to a DMA_HandleTypeDef structure as parameter.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_DMA_UnRegisterCallback

Function name	HAL_StatusTypeDef HAL_DMA_UnRegisterCallback (DMA_HandleTypeDef * hdma, HAL_DMA_CallbackIDTypeDef CallbackID)
Function description	UnRegister callbacks.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Stream. • CallbackID: User Callback identifier a HAL_DMA_CallbackIDTypeDef ENUM as parameter.

Return values • **HAL:** status

HAL_DMA_GetState

Function name **HAL_DMA_StateTypeDef HAL_DMA_GetState (DMA_HandleTypeDef * hdma)**

Function description Return the DMA handle state.

Parameters • **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.

Return values • **HAL:** state

HAL_DMA_GetError

Function name **uint32_t HAL_DMA_GetError (DMA_HandleTypeDef * hdma)**

Function description Return the DMA error code.

Parameters • **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.

Return values • **DMA:** Error Code

16.3 DMA Firmware driver defines

16.3.1 DMA

DMA Data transfer direction

DMA_PERIPH_TO_MEMORY	Peripheral to memory direction
DMA_MEMORY_TO_PERIPH	Memory to peripheral direction
DMA_MEMORY_TO_MEMORY	Memory to memory direction

DMA Error Code

HAL_DMA_ERROR_NONE	No error
HAL_DMA_ERROR_TE	Transfer error
HAL_DMA_ERROR_NO_XFER	no ongoing transfer
HAL_DMA_ERROR_TIMEOUT	Timeout error
HAL_DMA_ERROR_NOT_SUPPORTED	Not supported mode

DMA Exported Macros

_HAL_DMA_RESET_HANDLE_STATE	Description:
	• Reset DMA handle state.
	Parameters:
	• _HANDLE_ : DMA handle
	Return value:
	• None

`__HAL_DMA_ENABLE`

Description:

- Enable the specified DMA Channel.

Parameters:

- `__HANDLE__`: DMA handle

Return value:

- None

`__HAL_DMA_DISABLE`

Description:

- Disable the specified DMA Channel.

Parameters:

- `__HANDLE__`: DMA handle

Return value:

- None

`__HAL_DMA_GET_TC_FLAG_INDEX`

Description:

- Return the current DMA Channel transfer complete flag.

Parameters:

- `__HANDLE__`: DMA handle

Return value:

- The: specified transfer complete flag index.

Description:

- Return the current DMA Channel half transfer complete flag.

Parameters:

- `__HANDLE__`: DMA handle

Return value:

- The: specified half transfer complete flag index.

Description:

- Return the current DMA Channel transfer error flag.

Parameters:

- `__HANDLE__`: DMA handle

Return value:

- The: specified transfer error flag index.

Description:

- Return the current DMA Channel Global interrupt flag.

Parameters:

- __HANDLE__: DMA handle

Return value:

- The: specified transfer error flag index.

Description:

- Get the DMA Channel pending flags.

Parameters:

- __HANDLE__: DMA handle
- __FLAG__: Get the specified flag. This parameter can be any combination of the following values:
 - DMA_FLAG_TCx: Transfer complete flag
 - DMA_FLAG_HTx: Half transfer complete flag
 - DMA_FLAG_TEx: Transfer error flag
 - DMA_FLAG_GLx: Global interrupt flag Where x can be from 1 to 7 to select the DMA Channel x flag.

Return value:

- The: state of FLAG (SET or RESET).

Description:

- Clear the DMA Channel pending flags.

Parameters:

- __HANDLE__: DMA handle
- __FLAG__: specifies the flag to clear. This parameter can be any combination of the following values:
 - DMA_FLAG_TCx: Transfer complete flag
 - DMA_FLAG_HTx: Half transfer complete flag
 - DMA_FLAG_TEx: Transfer error flag
 - DMA_FLAG_GLx: Global interrupt flag Where x can be from 1 to 7 to select the DMA Channel x flag.

Return value:

- None

Description:

- Enable the specified DMA Channel interrupts.

Parameters:

- __HANDLE__: DMA handle
- __INTERRUPT__: specifies the DMA

interrupt sources to be enabled or disabled. This parameter can be any combination of the following values:

- DMA_IT_TC: Transfer complete interrupt mask
- DMA_IT_HT: Half transfer complete interrupt mask
- DMA_IT_TE: Transfer error interrupt mask

Return value:

- None

[__HAL_DMA_DISABLE_IT](#)

Description:

- Disable the specified DMA Channel interrupts.

Parameters:

- [__HANDLE__](#): DMA handle
- [__INTERRUPT__](#): specifies the DMA interrupt sources to be enabled or disabled. This parameter can be any combination of the following values:
 - DMA_IT_TC: Transfer complete interrupt mask
 - DMA_IT_HT: Half transfer complete interrupt mask
 - DMA_IT_TE: Transfer error interrupt mask

Return value:

- None

[__HAL_DMA_GET_IT_SOURCE](#)

Description:

- Check whether the specified DMA Channel interrupt is enabled or not.

Parameters:

- [__HANDLE__](#): DMA handle
- [__INTERRUPT__](#): specifies the DMA interrupt source to check. This parameter can be one of the following values:
 - DMA_IT_TC: Transfer complete interrupt mask
 - DMA_IT_HT: Half transfer complete interrupt mask
 - DMA_IT_TE: Transfer error interrupt mask

Return value:

- The: state of DMA_IT (SET or RESET).

[__HAL_DMA_GET_COUNTER](#)

Description:

- Return the number of remaining data units

in the current DMA Channel transfer.

Parameters:

- `__HANDLE__`: DMA handle

Return value:

- The: number of remaining data units in the current DMA Channel transfer.

DMA flag definitions

`DMA_FLAG_GL1`

`DMA_FLAG_TC1`

`DMA_FLAG_HT1`

`DMA_FLAG_TE1`

`DMA_FLAG_GL2`

`DMA_FLAG_TC2`

`DMA_FLAG_HT2`

`DMA_FLAG_TE2`

`DMA_FLAG_GL3`

`DMA_FLAG_TC3`

`DMA_FLAG_HT3`

`DMA_FLAG_TE3`

`DMA_FLAG_GL4`

`DMA_FLAG_TC4`

`DMA_FLAG_HT4`

`DMA_FLAG_TE4`

`DMA_FLAG_GL5`

`DMA_FLAG_TC5`

`DMA_FLAG_HT5`

`DMA_FLAG_TE5`

`DMA_FLAG_GL6`

`DMA_FLAG_TC6`

`DMA_FLAG_HT6`

`DMA_FLAG_TE6`

`DMA_FLAG_GL7`

`DMA_FLAG_TC7`

`DMA_FLAG_HT7`

`DMA_FLAG_TE7`

DMA interrupt enable definitions

`DMA_IT_TC`

DMA_IT_HT

DMA_IT_TE

DMA Memory data size

DMA_MDATAALIGN_BYTE Memory data alignment: Byte

DMA_MDATAALIGN_HALFWORD Memory data alignment: HalfWord

DMA_MDATAALIGN_WORD Memory data alignment: Word

DMA Memory incremented mode

DMA_MINC_ENABLE Memory increment mode Enable

DMA_MINC_DISABLE Memory increment mode Disable

DMA mode

DMA_NORMAL Normal mode

DMA_CIRCULAR Circular mode

DMA Peripheral data size

DMA_PDATAALIGN_BYTE Peripheral data alignment: Byte

DMA_PDATAALIGN_HALFWORD Peripheral data alignment: HalfWord

DMA_PDATAALIGN_WORD Peripheral data alignment: Word

DMA Peripheral incremented mode

DMA_PINC_ENABLE Peripheral increment mode Enable

DMA_PINC_DISABLE Peripheral increment mode Disable

DMA Priority level

DMA_PRIORITY_LOW Priority level : Low

DMA_PRIORITY_MEDIUM Priority level : Medium

DMA_PRIORITY_HIGH Priority level : High

DMA_PRIORITY VERY_HIGH Priority level : Very_High

DMA request

DMA_REQUEST_0

DMA_REQUEST_1

DMA_REQUEST_2

DMA_REQUEST_3

DMA_REQUEST_4

DMA_REQUEST_5

DMA_REQUEST_6

DMA_REQUEST_7

17 HAL FLASH Generic Driver

17.1 FLASH Firmware driver registers structures

17.1.1 FLASH_ProcessTypeDef

Data Fields

- `__IO FLASH_ProcedureTypeDef ProcedureOnGoing`
- `__IO uint32_t NbPagesToErase`
- `__IO uint32_t Address`
- `__IO uint32_t Page`
- `HAL_LockTypeDef Lock`
- `__IO uint32_t ErrorCode`

Field Documentation

- `__IO FLASH_ProcedureTypeDef FLASH_ProcessTypeDef::ProcedureOnGoing`
Internal variable to indicate which procedure is ongoing or not in IT context
- `__IO uint32_t FLASH_ProcessTypeDef::NbPagesToErase`
Internal variable to save the remaining sectors to erase in IT context
- `__IO uint32_t FLASH_ProcessTypeDef::Address`
Internal variable to save address selected for program or erase
- `__IO uint32_t FLASH_ProcessTypeDef::Page`
Internal variable to define the current page which is erasing
- `HAL_LockTypeDef FLASH_ProcessTypeDef::Lock`
FLASH locking object
- `__IO uint32_t FLASH_ProcessTypeDef::ErrorCode`
FLASH error code This parameter can be a value of [FLASH_Error_Codes](#)

17.2 FLASH Firmware driver API description

17.2.1 FLASH peripheral features

The Flash memory interface manages CPU AHB I-Code and D-Code accesses to the Flash memory. It implements the erase and program Flash memory operations and the read and write protection mechanisms.

The Flash memory interface accelerates code execution with a system of instruction prefetch.

The FLASH main features are:

- Flash memory read operations
- Flash memory program/erase operations
- Read / write protections
- Prefetch on I-Code
- Option Bytes programming

17.2.2 How to use this driver

This driver provides functions and macros to configure and program the FLASH memory of all STM32L1xx devices.

1. FLASH Memory I/O Programming functions: this group includes all needed functions to erase and program the main memory:

- Lock and Unlock the FLASH interface
 - Erase function: Erase page
 - Program functions: Fast Word and Half Page(should be executed from internal SRAM).
2. DATA EEPROM Programming functions: this group includes all needed functions to erase and program the DATA EEPROM memory:
- Lock and Unlock the DATA EEPROM interface.
 - Erase function: Erase Byte, erase HalfWord, erase Word, erase Double Word (should be executed from internal SRAM).
 - Program functions: Fast Program Byte, Fast Program Half-Word, FastProgramWord, Program Byte, Program Half-Word, Program Word and Program Double-Word (should be executed from internal SRAM).
3. FLASH Option Bytes Programming functions: this group includes all needed functions to manage the Option Bytes:
- Lock and Unlock the Option Bytes
 - Set/Reset the write protection
 - Set the Read protection Level
 - Program the user Option Bytes
 - Launch the Option Bytes loader
 - Set/Get the Read protection Level.
 - Set/Get the BOR level.
 - Get the Write protection.
 - Get the user option bytes.
4. Interrupts and flags management functions : this group includes all needed functions to:
- Handle FLASH interrupts
 - Wait for last FLASH operation according to its status
 - Get error flag status
5. FLASH Interface configuration functions: this group includes the management of following features:
- Enable/Disable the RUN PowerDown mode.
 - Enable/Disable the SLEEP PowerDown mode.
6. FLASH Peripheral State methods: this group includes the management of following features:
- Wait for the FLASH operation
 - Get the specific FLASH error flag

In addition to these function, this driver includes a set of macros allowing to handle the following operations:

- Set/Get the latency
- Enable/Disable the prefetch buffer
- Enable/Disable the 64 bit Read Access.
- Enable/Disable the Flash power-down
- Enable/Disable the FLASH interrupts
- Monitor the FLASH flags status

17.2.3 Programming operation functions

This subsection provides a set of functions allowing to manage the FLASH program operations.

The FLASH Memory Programming functions, includes the following functions:

- HAL_FLASH_Unlock(void);
- HAL_FLASH_Lock(void);

- HAL_FLASH_Program(uint32_t TypeProgram, uint32_t Address, uint32_t Data)
- HAL_FLASH_Program_IT(uint32_t TypeProgram, uint32_t Address, uint32_t Data)

Any operation of erase or program should follow these steps:

1. Call the HAL_FLASH_Unlock() function to enable the flash control register and program memory access.
2. Call the desired function to erase page or program data.
3. Call the HAL_FLASH_Lock() to disable the flash program memory access (recommended to protect the FLASH memory against possible unwanted operation).

17.2.4 Option Bytes Programming functions

The FLASH_Option Bytes Programming_functions, includes the following functions:

- HAL_FLASH_OB_Unlock(void);
- HAL_FLASH_OB_Lock(void);
- HAL_FLASH_OB_Launch(void);
- HAL_FLASHEx_OBProgram(FLASH_OBProgramInitTypeDef *pOBInit);
- HAL_FLASHEx_OBGetConfig(FLASH_OBProgramInitTypeDef *pOBInit);

Any operation of erase or program should follow these steps:

1. Call the HAL_FLASH_OB_Unlock() function to enable the Flash option control register access.
2. Call the following functions to program the desired option bytes.
 - HAL_FLASHEx_OBProgram(FLASH_OBProgramInitTypeDef *pOBInit);
3. Once all needed option bytes to be programmed are correctly written, call the HAL_FLASH_OB_Launch(void) function to launch the Option Bytes programming process.
4. Call the HAL_FLASH_OB_Lock() to disable the Flash option control register access (recommended to protect the option Bytes against possible unwanted operations).

Proprietary code Read Out Protection (PcROP):

1. The PcROP sector is selected by using the same option bytes as the Write protection. As a result, these 2 options are exclusive each other.
2. To activate PCROP mode for Flash sectors(s), you need to follow the sequence below:
 - Use this function HAL_FLASHEx_AdvOBProgram with PCROPState = OB_PCROP_STATE_ENABLE.

17.2.5 Peripheral Control functions

This subsection provides a set of functions allowing to control the FLASH memory operations.

This section contains the following APIs:

- [**HAL_FLASH_Unlock\(\)**](#)
- [**HAL_FLASH_Lock\(\)**](#)
- [**HAL_FLASH_OB_Unlock\(\)**](#)
- [**HAL_FLASH_OB_Lock\(\)**](#)
- [**HAL_FLASH_OB_Launch\(\)**](#)

17.2.6 Peripheral Errors functions

This subsection permit to get in run-time errors of the FLASH peripheral.

This section contains the following APIs:

- [*HAL_FLASH_GetError\(\)*](#)

17.2.7 Detailed description of functions

HAL_FLASH_Program

Function name	HAL_StatusTypeDef HAL_FLASH_Program (uint32_t TypeProgram, uint32_t Address, uint32_t Data)
Function description	Program word at a specified address.
Parameters	<ul style="list-style-type: none"> • TypeProgram: Indicate the way to program at a specified address. This parameter can be a value of FLASH Type Program • Address: Specifie the address to be programmed. • Data: Specifie the data to be programmed
Return values	<ul style="list-style-type: none"> • HAL_StatusTypeDef: HAL Status
Notes	<ul style="list-style-type: none"> • To correctly run this function, the HAL_FLASH_Unlock() function must be called before. Call the HAL_FLASH_Lock() to disable the flash memory access (recommended to protect the FLASH memory against possible unwanted operation).

HAL_FLASH_Program_IT

Function name	HAL_StatusTypeDef HAL_FLASH_Program_IT (uint32_t TypeProgram, uint32_t Address, uint32_t Data)
Function description	Program word at a specified address with interrupt enabled.
Parameters	<ul style="list-style-type: none"> • TypeProgram: Indicate the way to program at a specified address. This parameter can be a value of FLASH Type Program • Address: Specifie the address to be programmed. • Data: Specifie the data to be programmed
Return values	<ul style="list-style-type: none"> • HAL_StatusTypeDef: HAL Status

HAL_FLASH_IRQHandler

Function name	void HAL_FLASH_IRQHandler (void)
Function description	This function handles FLASH interrupt request.
Return values	<ul style="list-style-type: none"> • None:

HAL_FLASH_EndOfOperationCallback

Function name	void HAL_FLASH_EndOfOperationCallback (uint32_t ReturnValue)
Function description	FLASH end of operation interrupt callback.
Parameters	<ul style="list-style-type: none"> • ReturnValue: The value saved in this parameter depends on the ongoing procedure <ul style="list-style-type: none"> – Pages Erase: Address of the page which has been erased (if 0xFFFFFFFF, it means that all the selected pages have been erased)

- Program: Address which was selected for data program
- Return values
 - **none:**

HAL_FLASH_OperationErrorCallback

- | | |
|----------------------|--|
| Function name | void HAL_FLASH_OperationErrorCallback (uint32_t ReturnValue) |
| Function description | FLASH operation error interrupt callback. |
| Parameters | <ul style="list-style-type: none"> • ReturnValue: The value saved in this parameter depends on the ongoing procedure <ul style="list-style-type: none"> - Pages Erase: Address of the page which returned an error - Program: Address which was selected for data program |
| Return values | <ul style="list-style-type: none"> • none: |

HAL_FLASH_Unlock

- | | |
|----------------------|--|
| Function name | HAL_StatusTypeDef HAL_FLASH_Unlock (void) |
| Function description | Unlock the FLASH control register access. |
| Return values | <ul style="list-style-type: none"> • HAL: Status |

HAL_FLASH_Lock

- | | |
|----------------------|--|
| Function name | HAL_StatusTypeDef HAL_FLASH_Lock (void) |
| Function description | Locks the FLASH control register access. |
| Return values | <ul style="list-style-type: none"> • HAL: Status |

HAL_FLASH_OB_Unlock

- | | |
|----------------------|--|
| Function name | HAL_StatusTypeDef HAL_FLASH_OB_Unlock (void) |
| Function description | Unlock the FLASH Option Control Registers access. |
| Return values | <ul style="list-style-type: none"> • HAL: Status |

HAL_FLASH_OB_Lock

- | | |
|----------------------|--|
| Function name | HAL_StatusTypeDef HAL_FLASH_OB_Lock (void) |
| Function description | Lock the FLASH Option Control Registers access. |
| Return values | <ul style="list-style-type: none"> • HAL: Status |

HAL_FLASH_OB_Launch

- | | |
|----------------------|---|
| Function name | HAL_StatusTypeDef HAL_FLASH_OB_Launch (void) |
| Function description | Launch the option byte loading. |
| Return values | <ul style="list-style-type: none"> • HAL: Status |
| Notes | <ul style="list-style-type: none"> • This function will reset automatically the MCU. |

HAL_FLASH_GetError

Function name	<code>uint32_t HAL_FLASH_GetError (void)</code>
Function description	Get the specific FLASH error flag.
Return values	<ul style="list-style-type: none"> • FLASH_ErrorCode: The returned value can be: FLASH Error Codes

FLASH_WaitForLastOperation

Function name	<code>HAL_StatusTypeDef FLASH_WaitForLastOperation (uint32_t Timeout)</code>
Function description	Wait for a FLASH operation to complete.
Parameters	<ul style="list-style-type: none"> • Timeout: maximum flash operation timeout
Return values	<ul style="list-style-type: none"> • HAL: Status

17.3 FLASH Firmware driver defines

17.3.1 FLASH

FLASH Error Codes

<code>HAL_FLASH_ERROR_NONE</code>	No error
<code>HAL_FLASH_ERROR_PGA</code>	Programming alignment error
<code>HAL_FLASH_ERROR_WRP</code>	Write protection error
<code>HAL_FLASH_ERROR_OPTV</code>	Option validity error
<code>HAL_FLASH_ERROR_SIZE</code>	
<code>HAL_FLASH_ERROR_RD</code>	Read protected error
<code>HAL_FLASH_ERROR_OPTVUSR</code>	Option UserValidity Error.
<code>HAL_FLASH_ERROR_OPERATION</code>	Not used

FLASH Flags

<code>FLASH_FLAG_BSY</code>	FLASH Busy flag
<code>FLASH_FLAG_EOP</code>	FLASH End of Programming flag
<code>FLASH_FLAG_ENDHV</code>	FLASH End of High Voltage flag
<code>FLASH_FLAG_READY</code>	FLASH Ready flag after low power mode
<code>FLASH_FLAG_WRPERR</code>	FLASH Write protected error flag
<code>FLASH_FLAG_PGAERR</code>	FLASH Programming Alignment error flag
<code>FLASH_FLAG_SIZERR</code>	FLASH Size error flag
<code>FLASH_FLAG_OPTVERR</code>	FLASH Option Validity error flag
<code>FLASH_FLAG_OPTVERRUSR</code>	FLASH Option User Validity error flag

FLASH Interrupts

<code>_HAL_FLASH_ENABLE_IT</code>	Description:
	<ul style="list-style-type: none"> • Enable the specified FLASH interrupt.

Parameters:

- __INTERRUPT__: FLASH interrupt This parameter can be any combination of the following values:
 - FLASH_IT_EOP End of FLASH Operation Interrupt
 - FLASH_IT_ERR Error Interrupt

Return value:

- none

[__HAL_FLASH_DISABLE_IT](#)**Description:**

- Disable the specified FLASH interrupt.

Parameters:

- __INTERRUPT__: FLASH interrupt This parameter can be any combination of the following values:
 - FLASH_IT_EOP End of FLASH Operation Interrupt
 - FLASH_IT_ERR Error Interrupt

Return value:

- none

[__HAL_FLASH_GET_FLAG](#)**Description:**

- Get the specified FLASH flag status.

Parameters:

- __FLAG__: specifies the FLASH flag to check. This parameter can be one of the following values:
 - FLASH_FLAG_BSY FLASH Busy flag
 - FLASH_FLAG_EOP FLASH End of Operation flag
 - FLASH_FLAG_ENDHV FLASH End of High Voltage flag
 - FLASH_FLAG_READY FLASH Ready flag after low power mode
 - FLASH_FLAG_PGAERR FLASH Programming Alignment error flag
 - FLASH_FLAG_SIZERR FLASH Size error flag
 - FLASH_FLAG_OPTVERR FLASH Option validity error flag
 - FLASH_FLAG_OPTVERRUSR FLASH Option User validity error
 - FLASH_FLAG_WRPERR FLASH Write protected error flag

Return value:

- The: new state of __FLAG__ (SET or RESET).

[__HAL_FLASH_CLEAR_FLAG](#)**Description:**

- Clear the specified FLASH flag.

Parameters:

- `__FLAG__`: specifies the FLASH flags to clear. This parameter can be any combination of the following values:
 - `FLASH_FLAG_EOP` FLASH End of Operation flag
 - `FLASH_FLAG_PGAERR` FLASH Programming Alignment error flag
 - `FLASH_FLAG_SIZERR` FLASH Size error flag
 - `FLASH_FLAG_OPTVERR` FLASH Option validity error error flag
 - `FLASH_FLAG_OPTVERRUSR` FLASH Option User validity error
 - `FLASH_FLAG_WRPERR` FLASH Write protected error flag

Return value:

- none

FLASH Interrupts

`FLASH_IT_EOP` End of programming interrupt source

`FLASH_IT_ERR` Error interrupt source

FLASH Keys

`FLASH_PDKEY1` Flash power down key1

`FLASH_PDKEY2` Flash power down key2: used with `FLASH_PDKEY1` to unlock the `RUN_PD` bit in `FLASH_ACR`

`FLASH_PEKEY1` Flash program erase key1

`FLASH_PEKEY2` Flash program erase key: used with `FLASH_PEKEY2` to unlock the write access to the `FLASH_PECR` register and data EEPROM

`FLASH_PRGKEY1` Flash program memory key1

`FLASH_PRGKEY2` Flash program memory key2: used with `FLASH_PRGKEY2` to unlock the program memory

`FLASH_OPTKEY1` Flash option key1

`FLASH_OPTKEY2` Flash option key2: used with `FLASH_OPTKEY1` to unlock the write access to the option byte block

FLASH Latency

`FLASH_LATENCY_0` FLASH Zero Latency cycle

`FLASH_LATENCY_1` FLASH One Latency cycle

FLASH size information

`FLASH_SIZE`

`FLASH_PAGE_SIZE` FLASH Page Size in bytes

FLASH Type Program

`FLASH_TYPEPROGRAM_WORD` Program a word (32-bit) at a specified address.

18 HAL FLASH Extension Driver

18.1 FLASHEx Firmware driver registers structures

18.1.1 FLASH_EraseInitTypeDef

Data Fields

- *uint32_t TypeErase*
- *uint32_t PageAddress*
- *uint32_t NbPages*

Field Documentation

- *uint32_t FLASH_EraseInitTypeDef::TypeErase*
TypeErase: Page Erase only. This parameter can be a value of
FLASHEx_Type_Erase
- *uint32_t FLASH_EraseInitTypeDef::PageAddress*
PageAddress: Initial FLASH address to be erased This parameter must be a value belonging to FLASH Programm address (depending on the devices)
- *uint32_t FLASH_EraseInitTypeDef::NbPages*
NbPages: Number of pages to be erased. This parameter must be a value between 1 and (max number of pages - value of Initial page)

18.1.2 FLASH_OBProgramInitTypeDef

Data Fields

- *uint32_t OptionType*
- *uint32_t WRPState*
- *uint32_t WRPSector0To31*
- *uint32_t WRPSector32To63*
- *uint32_t WRPSector64To95*
- *uint8_t RDPLevel*
- *uint8_t BORLevel*
- *uint8_t USERConfig*

Field Documentation

- *uint32_t FLASH_OBProgramInitTypeDef::OptionType*
OptionType: Option byte to be configured. This parameter can be a value of
FLASHEx_Option_Type
- *uint32_t FLASH_OBProgramInitTypeDef::WRPState*
WRPState: Write protection activation or deactivation. This parameter can be a value of
FLASHEx_WRP_State
- *uint32_t FLASH_OBProgramInitTypeDef::WRPSector0To31*
WRPSector0To31: specifies the sector(s) which are write protected between Sector 0 to 31 This parameter can be a combination of
FLASHEx_Option_Bytess_Write_Protection1
- *uint32_t FLASH_OBProgramInitTypeDef::WRPSector32To63*
WRPSector32To63: specifies the sector(s) which are write protected between Sector 32 to 63 This parameter can be a combination of
FLASHEx_Option_Bytess_Write_Protection2
- *uint32_t FLASH_OBProgramInitTypeDef::WRPSector64To95*
WRPSector64to95: specifies the sector(s) which are write protected between Sector

- 64 to 95 This parameter can be a combination of [**FLASHEx_Option_Bytes_Write_Protection3**](#)
- ***uint8_t FLASH_OBProgramInitTypeDef::RDPLevel***
RDPLevel: Set the read protection level. This parameter can be a value of [**FLASHEx_Option_Bytes_Read_Protection**](#)
 - ***uint8_t FLASH_OBProgramInitTypeDef::BORLevel***
BORLevel: Set the BOR Level. This parameter can be a value of [**FLASHEx_Option_Bytes_BOR_Level**](#)
 - ***uint8_t FLASH_OBProgramInitTypeDef::USERConfig***
USERConfig: Program the FLASH User Option Byte: IWDG_SW / RST_STOP / RST_STDBY. This parameter can be a combination of [**FLASHEx_Option_Bytes_IWWatchdog**](#), [**FLASHEx_Option_Bytes_nRST_STOP**](#) and [**FLASHEx_Option_Bytes_nRST_STDBY**](#)

18.1.3 **FLASH_AdvOBProgramInitTypeDef**

Data Fields

- ***uint32_t OptionType***
- ***uint16_t BootConfig***

Field Documentation

- ***uint32_t FLASH_AdvOBProgramInitTypeDef::OptionType***
OptionType: Option byte to be configured for extension . This parameter can be a value of [**FLASHEx_OptionAdv_Type**](#)
- ***uint16_t FLASH_AdvOBProgramInitTypeDef::BootConfig***
BootConfig: specifies Option bytes for boot config This parameter can be a value of [**FLASHEx_Option_Bytes_BOOT**](#)

18.2 **FLASHEx Firmware driver API description**

18.2.1 **FLASH Erasing Programming functions**

The FLASH Memory Erasing functions, includes the following functions:

- @ref HAL_FLASHEx_Erase: return only when erase has been done
- @ref HAL_FLASHEx_Erase_IT: end of erase is done when @ref HAL_FLASH_EndOfOperationCallback is called with parameter 0xFFFFFFFF

Any operation of erase should follow these steps:

1. Call the @ref HAL_FLASH_Unlock() function to enable the flash control register and program memory access.
2. Call the desired function to erase page.
3. Call the @ref HAL_FLASH_Lock() to disable the flash program memory access (recommended to protect the FLASH memory against possible unwanted operation).

This section contains the following APIs:

- [**HAL_FLASHEx_Erase\(\)**](#)
- [**HAL_FLASHEx_Erase_IT\(\)**](#)

18.2.2 **Option Bytes Programming functions**

Any operation of erase or program should follow these steps:

1. Call the @ref HAL_FLASH_OB_Unlock() function to enable the Flash option control register access.
2. Call following function to program the desired option bytes.

- @ref HAL_FLASHEx_OBProgram: - To Enable/Disable the desired sector write protection. - To set the desired read Protection Level. - To configure the user option Bytes: IWDG, STOP and the Standby. - To Set the BOR level.
- 3. Once all needed option bytes to be programmed are correctly written, call the @ref HAL_FLASH_OB_Launch(void) function to launch the Option Bytes programming process.
- 4. Call the @ref HAL_FLASH_OB_Lock() to disable the Flash option control register access (recommended to protect the option Bytes against possible unwanted operations).

Proprietary code Read Out Protection (PcROP):

1. The PcROP sector is selected by using the same option bytes as the Write protection (nWRPi bits). As a result, these 2 options are exclusive each other.
2. In order to activate the PcROP (change the function of the nWRPi option bits), the SPRMOD option bit must be activated.
3. The active value of nWRPi bits is inverted when PCROP mode is active, this means: if SPRMOD = 1 and nWRPi = 1 (default value), then the user sector "i" is read/write protected.
4. To activate PCROP mode for Flash sector(s), you need to call the following function:
 - @ref HAL_FLASHEx_AdvOBProgram in selecting sectors to be read/write protected
 - @ref HAL_FLASHEx_OB_SelectPCROP to enable the read/write protection
5. PcROP is available only in STM32L151xBA, STM32L152xBA, STM32L151xC, STM32L152xC & STM32L162xC devices.

This section contains the following APIs:

- [**HAL_FLASHEx_OBProgram\(\)**](#)
- [**HAL_FLASHEx_OBGetConfig\(\)**](#)
- [**HAL_FLASHEx_AdvOBProgram\(\)**](#)
- [**HAL_FLASHEx_AdvOBGetConfig\(\)**](#)

18.2.3 DATA EEPROM Programming functions

Any operation of erase or program should follow these steps:

1. Call the @ref HAL_FLASHEx_DATAEEPROM_Unlock() function to enable the data EEPROM access and Flash program erase control register access.
2. Call the desired function to erase or program data.
3. Call the @ref HAL_FLASHEx_DATAEEPROM_Lock() to disable the data EEPROM access and Flash program erase control register access(recommended to protect the DATA_EEPROM against possible unwanted operation).

This section contains the following APIs:

- [**HAL_FLASHEx_DATAEEPROM_Unlock\(\)**](#)
- [**HAL_FLASHEx_DATAEEPROM_Lock\(\)**](#)
- [**HAL_FLASHEx_DATAEEPROM_Erase\(\)**](#)
- [**HAL_FLASHEx_DATAEEPROM_Program\(\)**](#)
- [**HAL_FLASHEx_DATAEEPROM_EnableFixedTimeProgram\(\)**](#)
- [**HAL_FLASHEx_DATAEEPROM_DisableFixedTimeProgram\(\)**](#)

18.2.4 Detailed description of functions

HAL_FLASHEx_Erase

Function name

HAL_StatusTypeDef HAL_FLASHEx_Erase

(FLASH_EraselInitTypeDef * pEraselInit, uint32_t * PageError)

Function description	Erase the specified FLASH memory Pages.
Parameters	<ul style="list-style-type: none"> • pEraselInit: pointer to an FLASH_EraselInitTypeDef structure that contains the configuration information for the erasing. • PageError: pointer to variable that contains the configuration information on faulty page in case of error (0xFFFFFFFF means that all the pages have been correctly erased)
Return values	<ul style="list-style-type: none"> • HAL_StatusTypeDef: HAL Status
Notes	<ul style="list-style-type: none"> • To correctly run this function, the HAL_FLASH_Unlock() function must be called before. Call the HAL_FLASH_Lock() to disable the flash memory access (recommended to protect the FLASH memory against possible unwanted operation) • For STM32L151xDX/STM32L152xDX/STM32L162xDX, as memory is not continuous between 2 banks, user should perform pages erase by bank only.

HAL_FLASHEx_Erase_IT

Function name	HAL_StatusTypeDef HAL_FLASHEx_Erase_IT (FLASH_EraselInitTypeDef * pEraselInit)
Function description	Perform a page erase of the specified FLASH memory pages with interrupt enabled.
Parameters	<ul style="list-style-type: none"> • pEraselInit: pointer to an FLASH_EraselInitTypeDef structure that contains the configuration information for the erasing.
Return values	<ul style="list-style-type: none"> • HAL_StatusTypeDef: HAL Status
Notes	<ul style="list-style-type: none"> • To correctly run this function, the HAL_FLASH_Unlock() function must be called before. Call the HAL_FLASH_Lock() to disable the flash memory access (recommended to protect the FLASH memory against possible unwanted operation) End of erase is done when HAL_FLASH_EndOfOperationCallback is called with parameter 0xFFFFFFFF • For STM32L151xDX/STM32L152xDX/STM32L162xDX, as memory is not continuous between 2 banks, user should perform pages erase by bank only.

HAL_FLASHEx_OBProgram

Function name	HAL_StatusTypeDef HAL_FLASHEx_OBProgram (FLASH_OBProgramInitTypeDef * pOBInit)
Function description	Program option bytes.
Parameters	<ul style="list-style-type: none"> • pOBInit: pointer to an FLASH_OBInitStruct structure that contains the configuration information for the programming.
Return values	<ul style="list-style-type: none"> • HAL_StatusTypeDef: HAL Status

HAL_FLASHEx_OBGetConfig

Function name	void HAL_FLASHEx_OBGetConfig
---------------	-------------------------------------

(FLASH_OBProgramInitTypeDef * pOBInit)

Function description	Get the Option byte configuration.
Parameters	<ul style="list-style-type: none"> • pOBInit: pointer to an FLASH_OBInitStruct structure that contains the configuration information for the programming.
Return values	<ul style="list-style-type: none"> • None:

HAL_FLASHEx_AdvOBProgram

Function name	HAL_StatusTypeDef HAL_FLASHEx_AdvOBProgram (FLASH_AdvOBProgramInitTypeDef * pAdvOBInit)
Function description	Program option bytes.
Parameters	<ul style="list-style-type: none"> • pAdvOBInit: pointer to an FLASH_AdvOBProgramInitTypeDef structure that contains the configuration information for the programming.
Return values	<ul style="list-style-type: none"> • HAL_StatusTypeDef: HAL Status
Notes	<ul style="list-style-type: none"> • This function can be used only for Cat2 & Cat3 devices for PCROP and Cat4 & Cat5 for BFB2.

HAL_FLASHEx_AdvOBGetConfig

Function name	void HAL_FLASHEx_AdvOBGetConfig (FLASH_AdvOBProgramInitTypeDef * pAdvOBInit)
Function description	Get the OBEX byte configuration.
Parameters	<ul style="list-style-type: none"> • pAdvOBInit: pointer to an FLASH_AdvOBProgramInitTypeDef structure that contains the configuration information for the programming.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This function can be used only for Cat2 & Cat3 devices for PCROP and Cat4 & Cat5 for BFB2.

HAL_FLASHEx_DATAEEPROM_Unlock

Function name	HAL_StatusTypeDef HAL_FLASHEx_DATAEEPROM_Unlock (void)
Function description	Unlocks the data memory and FLASH_PECR register access.
Return values	<ul style="list-style-type: none"> • HAL_StatusTypeDef: HAL Status

HAL_FLASHEx_DATAEEPROM_Lock

Function name	HAL_StatusTypeDef HAL_FLASHEx_DATAEEPROM_Lock (void)
Function description	Locks the Data memory and FLASH_PECR register access.
Return values	<ul style="list-style-type: none"> • HAL_StatusTypeDef: HAL Status

HAL_FLASHEx_DATAEEPROM_Erase

Function name	HAL_StatusTypeDef HAL_FLASHEx_DATAEEPROM_Erase (uint32_t TypeErase, uint32_t Address)
Function description	Erase a word in data memory.
Parameters	<ul style="list-style-type: none"> • Address: specifies the address to be erased. • TypeErase: Indicate the way to erase at a specified address. This parameter can be a value of FLASH Type Program
Return values	• HAL_StatusTypeDef: HAL Status
Notes	<ul style="list-style-type: none"> • To correctly run this function, the HAL_FLASHEx_DATAEEPROM_Unlock() function must be called before. Call the HAL_FLASHEx_DATAEEPROM_Lock() to the data EEPROM access and Flash program erase control register access(recommended to protect the DATA_EEPROM against possible unwanted operation).

HAL_FLASHEx_DATAEEPROM_Program

Function name	HAL_StatusTypeDef HAL_FLASHEx_DATAEEPROM_Program (uint32_t TypeProgram, uint32_t Address, uint32_t Data)
Function description	Program word at a specified address.
Parameters	<ul style="list-style-type: none"> • TypeProgram: Indicate the way to program at a specified address. This parameter can be a value of FLASHEx Type Program Data • Address: specifie the address to be programmed. • Data: specifie the data to be programmed
Return values	• HAL_StatusTypeDef: HAL Status
Notes	<ul style="list-style-type: none"> • To correctly run this function, the HAL_FLASHEx_DATAEEPROM_Unlock() function must be called before. Call the HAL_FLASHEx_DATAEEPROM_Unlock() to he data EEPROM access and Flash program erase control register access(recommended to protect the DATA_EEPROM against possible unwanted operation). • The function HAL_FLASHEx_DATAEEPROM_EnableFixedTimeProgram() can be called before this function to configure the Fixed Time Programming.

HAL_FLASHEx_DATAEEPROM_EnableFixedTimeProgram

Function name	void HAL_FLASHEx_DATAEEPROM_EnableFixedTimeProgram (void)
Function description	Enable DATA EEPROM fixed Time programming (2*Tprog).
Return values	<ul style="list-style-type: none"> • None:

HAL_FLASHEx_DATAEEPROM_DisableFixedTimeProgram

Function name **void HAL_FLASHEx_DATAEEPROM_DisableFixedTimeProgram (void)**

Function description Disables DATA EEPROM fixed Time programming (2*Tprog).

Return values • **None:**

18.3 FLASHEx Firmware driver defines

18.3.1 FLASHEx

FLASHEx Address

IS_FLASH_DATA_ADDRESS

IS_FLASH_PROGRAM_ADDRESS

IS_FLASH_PROGRAM_BANK1_ADDRESS

IS_FLASH_PROGRAM_BANK2_ADDRESS

IS_NBPAGES

FLASHEx Exported Macros

_HAL_FLASH_SET_LATENCY

Description:

- Set the FLASH Latency.

Parameters:

- **_LATENCY_**: FLASH Latency This parameter can be one of the following values:
 - FLASH_LATENCY_0 FLASH Zero Latency cycle
 - FLASH_LATENCY_1 FLASH One Latency cycle

Return value:

- none

_HAL_FLASH_GET_LATENCY

Description:

- Get the FLASH Latency.

Return value:

- FLASH: Latency This parameter can be one of the following values:
 - FLASH_LATENCY_0 FLASH Zero Latency cycle
 - FLASH_LATENCY_1 FLASH One Latency cycle

_HAL_FLASH_ACC64_ENABLE

Description:

- Enable the FLASH 64-bit access.

Return value:

- none

Notes:

- Read access 64 bit is used. This bit cannot be written at the same time as the LATENCY and PRFTEN bits.

`__HAL_FLASH_ACC64_DISABLE`

Description:

- Disable the FLASH 64-bit access.

Return value:

- none

Notes:

- Read access 32 bit is used To reset this bit, the LATENCY should be zero wait state and the prefetch off.

`__HAL_FLASH_PREFETCH_BUFFER_ENABLE`

Description:

- Enable the FLASH prefetch buffer.

Return value:

- none

`__HAL_FLASH_PREFETCH_BUFFER_DISABLE`

Description:

- Disable the FLASH prefetch buffer.

Return value:

- none

`__HAL_FLASH_SLEEP_POWERDOWN_ENABLE`

Description:

- Enable the FLASH power down during Sleep mode.

Return value:

- none

`__HAL_FLASH_SLEEP_POWERDOWN_DISABLE`

Description:

- Disable the FLASH power down during Sleep mode.

Return value:

- none

`__HAL_FLASH_POWER_DOWN_ENABLE`

Notes:

- Writing this bit to 0 this bit, automatically the keys are loss and a new unlock sequence is

necessary to re-write it to 1.

__HAL_FLASH_POWER_DOWN_DISABLE

Notes:

- Writing this bit to 0 this bit, automatically the keys are loss and a new unlock sequence is necessary to re-write it to 1.

FLASHEx Option Advanced Type

OPTIONBYTE_BOOTCONFIG BOOTConfig option byte configuration

FLASHEx Option Bytes BOOT

OB_BOOT_BANK2	At startup, if boot pins are set in boot from user Flash position and this parameter is selected the device will boot from Bank 2 or Bank 1, depending on the activation of the bank
OB_BOOT_BANK1	At startup, if boot pins are set in boot from user Flash position and this parameter is selected the device will boot from Bank1(Default)

FLASHEx Option Bytes BOR Level

OB_BOR_OFF	BOR is disabled at power down, the reset is asserted when the VDD power supply reaches the PDR(Power Down Reset) threshold (1.5V)
OB_BOR_LEVEL1	BOR Reset threshold levels for 1.7V - 1.8V VDD power supply
OB_BOR_LEVEL2	BOR Reset threshold levels for 1.9V - 2.0V VDD power supply
OB_BOR_LEVEL3	BOR Reset threshold levels for 2.3V - 2.4V VDD power supply
OB_BOR_LEVEL4	BOR Reset threshold levels for 2.55V - 2.65V VDD power supply
OB_BOR_LEVEL5	BOR Reset threshold levels for 2.8V - 2.9V VDD power supply

FLASHEx Option Bytes IWatchdog

OB_IWDG_SW	Software WDG selected
OB_IWDG_HW	Hardware WDG selected

FLASHEx Option Bytes nRST_STDBY

OB_STDBY_NORST	No reset generated when entering in STANDBY
OB_STDBY_RST	Reset generated when entering in STANDBY

FLASHEx Option Bytes nRST_STOP

OB_STOP_NORST	No reset generated when entering in STOP
OB_STOP_RST	Reset generated when entering in STOP

FLASHEx Option Bytes Read Protection

OB_RDP_LEVEL_0	
OB_RDP_LEVEL_1	
OB_RDP_LEVEL_2	

FLASHEx Option Bytes Write Protection1

OB_WRP1_PAGES0TO15
OB_WRP1_PAGES16TO31
OB_WRP1_PAGES32TO47
OB_WRP1_PAGES48TO63
OB_WRP1_PAGES64TO79
OB_WRP1_PAGES80TO95
OB_WRP1_PAGES96TO111
OB_WRP1_PAGES112TO127
OB_WRP1_PAGES128TO143
OB_WRP1_PAGES144TO159
OB_WRP1_PAGES160TO175
OB_WRP1_PAGES176TO191
OB_WRP1_PAGES192TO207
OB_WRP1_PAGES208TO223
OB_WRP1_PAGES224TO239
OB_WRP1_PAGES240TO255
OB_WRP1_PAGES256TO271
OB_WRP1_PAGES272TO287
OB_WRP1_PAGES288TO303
OB_WRP1_PAGES304TO319
OB_WRP1_PAGES320TO335
OB_WRP1_PAGES336TO351
OB_WRP1_PAGES352TO367
OB_WRP1_PAGES368TO383
OB_WRP1_PAGES384TO399
OB_WRP1_PAGES400TO415
OB_WRP1_PAGES416TO431
OB_WRP1_PAGES432TO447
OB_WRP1_PAGES448TO463
OB_WRP1_PAGES464TO479
OB_WRP1_PAGES480TO495
OB_WRP1_PAGES496TO511
OB_WRP1_ALLPAGES Write protection of all Sectors
FLASHEx Option Bytes Write Protection2
OB_WRP2_PAGES512TO527
OB_WRP2_PAGES528TO543

OB_WRP2_PAGES544TO559
OB_WRP2_PAGES560TO575
OB_WRP2_PAGES576TO591
OB_WRP2_PAGES592TO607
OB_WRP2_PAGES608TO623
OB_WRP2_PAGES624TO639
OB_WRP2_PAGES640TO655
OB_WRP2_PAGES656TO671
OB_WRP2_PAGES672TO687
OB_WRP2_PAGES688TO703
OB_WRP2_PAGES704TO719
OB_WRP2_PAGES720TO735
OB_WRP2_PAGES736TO751
OB_WRP2_PAGES752TO767
OB_WRP2_PAGES768TO783
OB_WRP2_PAGES784TO799
OB_WRP2_PAGES800TO815
OB_WRP2_PAGES816TO831
OB_WRP2_PAGES832TO847
OB_WRP2_PAGES848TO863
OB_WRP2_PAGES864TO879
OB_WRP2_PAGES880TO895
OB_WRP2_PAGES896TO911
OB_WRP2_PAGES912TO927
OB_WRP2_PAGES928TO943
OB_WRP2_PAGES944TO959
OB_WRP2_PAGES960TO975
OB_WRP2_PAGES976TO991
OB_WRP2_PAGES992TO1007
OB_WRP2_PAGES1008TO1023
OB_WRP2_ALLPAGES

FLASHEx Option Bytes Write Protection3

OB_WRP3_PAGES1024TO1039
OB_WRP3_PAGES1040TO1055
OB_WRP3_PAGES1056TO1071
OB_WRP3_PAGES1072TO1087

OB_WRP3_PAGES1088TO1103
OB_WRP3_PAGES1104TO1119
OB_WRP3_PAGES1120TO1135
OB_WRP3_PAGES1136TO1151
OB_WRP3_PAGES1152TO1167
OB_WRP3_PAGES1168TO1183
OB_WRP3_PAGES1184TO1199
OB_WRP3_PAGES1200TO1215
OB_WRP3_PAGES1216TO1231
OB_WRP3_PAGES1232TO1247
OB_WRP3_PAGES1248TO1263
OB_WRP3_PAGES1264TO1279
OB_WRP3_PAGES1280TO1295
OB_WRP3_PAGES1296TO1311
OB_WRP3_PAGES1312TO1327
OB_WRP3_PAGES1328TO1343
OB_WRP3_PAGES1344TO1359
OB_WRP3_PAGES1360TO1375
OB_WRP3_PAGES1376TO1391
OB_WRP3_PAGES1392TO1407
OB_WRP3_PAGES1408TO1423
OB_WRP3_PAGES1424TO1439
OB_WRP3_PAGES1440TO1455
OB_WRP3_PAGES1456TO1471
OB_WRP3_PAGES1472TO1487
OB_WRP3_PAGES1488TO1503
OB_WRP3_PAGES1504TO1519
OB_WRP3_PAGES1520TO1535
OB_WRP3_ALLPAGES Write protection of all Sectors

FLASHEx Option Type

OPTIONBYTE_WRP	WRP option byte configuration
OPTIONBYTE_RDP	RDP option byte configuration
OPTIONBYTE_USER	USER option byte configuration
OPTIONBYTE_BOR	BOR option byte configuration

FLASHEx_Type_Erase

FLASH_TYPEERASE_PAGES Page erase only

FLASHEx Type Erase Data

<code>FLASH_TYPEERASEDATA_BYTE</code>	Erase byte (8-bit) at a specified address.
<code>FLASH_TYPEERASEDATA_HALFWORD</code>	Erase a half-word (16-bit) at a specified address.
<code>FLASH_TYPEERASEDATA_WORD</code>	Erase a word (32-bit) at a specified address.
<i>FLASHEx Type Program Data</i>	
<code>FLASH_TYPEPROGRAMDATA_BYTE</code>	Program byte (8-bit) at a specified address.
<code>FLASH_TYPEPROGRAMDATA_HALFWORD</code>	Program a half-word (16-bit) at a specified address.
<code>FLASH_TYPEPROGRAMDATA_WORD</code>	Program a word (32-bit) at a specified address.
<code>FLASH_TYPEPROGRAMDATA_FASTBYTE</code>	Fast Program byte (8-bit) at a specified address.
<code>FLASH_TYPEPROGRAMDATA_FASTHALFWORD</code>	Fast Program a half-word (16-bit) at a specified address.
<code>FLASH_TYPEPROGRAMDATA_FASTWORD</code>	Fast Program a word (32-bit) at a specified address.
<i>FLASHEx WRP State</i>	
<code>OB_WRPSTATE_DISABLE</code>	Disable the write protection of the desired sectors
<code>OB_WRPSTATE_ENABLE</code>	Enable the write protection of the desired sectors

19 HAL FLASH__RAMFUNC Generic Driver

19.1 FLASH__RAMFUNC Firmware driver API description

19.1.1 Peripheral errors functions

This subsection permit to get in run-time errors of the FLASH peripheral.

This section contains the following APIs:

- [*HAL_FLASHEx_GetError\(\)*](#)

19.1.2 Detailed description of functions

HAL_FLASHEx_EnableRunPowerDown

Function name [__RAM_FUNC HAL_FLASHEx_EnableRunPowerDown \(void \)](#)

Function description Enable the power down mode during RUN mode.

Return values • **HAL:** status

Notes • This function can be used only when the user code is running from Internal SRAM.

HAL_FLASHEx_DisableRunPowerDown

Function name [__RAM_FUNC HAL_FLASHEx_DisableRunPowerDown \(void \)](#)

Function description Disable the power down mode during RUN mode.

Return values • **HAL:** status

Notes • This function can be used only when the user code is running from Internal SRAM.

HAL_FLASHEx_EraseParallelPage

Function name [__RAM_FUNC HAL_FLASHEx_EraseParallelPage \(uint32_t Page_Address1, uint32_t Page_Address2\)](#)

Function description Erases a specified 2 pages in program memory in parallel.

Parameters

- **Page_Address1:** The page address in program memory to be erased in the first Bank (BANK1). This parameter should be between FLASH_BASE and FLASH_BANK1_END.
- **Page_Address2:** The page address in program memory to be erased in the second Bank (BANK2). This parameter should be between FLASH_BANK2_BASE and FLASH_BANK2_END.

Return values • **HAL:** status

Notes • This function can be used only for STM32L151xD, STM32L152xD, STM32L162xD and Cat5 devices. To correctly run this function, the HAL_FLASH_Unlock() function must be called before. Call the HAL_FLASH_Lock() to disable the flash memory access (recommended to protect the

- FLASH memory against possible unwanted operation).
- A Page is erased in the Program memory only if the address to load is the start address of a page (multiple of FLASH_PAGE_SIZE bytes).

HAL_FLASHEx_ProgramParallelHalfPage

Function name	<code>__RAM_FUNC HAL_FLASHEx_ProgramParallelHalfPage (uint32_t Address1, uint32_t * pBuffer1, uint32_t Address2, uint32_t * pBuffer2)</code>
Function description	Program 2 half pages in program memory in parallel (half page size is 32 Words).
Parameters	<ul style="list-style-type: none"> • Address1: specifies the first address to be written in the first bank (BANK1). This parameter should be between FLASH_BASE and (FLASH_BANK1_END - FLASH_PAGE_SIZE). • pBuffer1: pointer to the buffer containing the data to be written to the first half page in the first bank. • Address2: specifies the second address to be written in the second bank (BANK2). This parameter should be between FLASH_BANK2_BASE and (FLASH_BANK2_END - FLASH_PAGE_SIZE). • pBuffer2: pointer to the buffer containing the data to be written to the second half page in the second bank.
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • This function can be used only for STM32L151xD, STM32L152xD, STM32L162xD and Cat5 devices. • To correctly run this function, the HAL_FLASH_Unlock() function must be called before. Call the HAL_FLASH_Lock() to disable the flash memory access (recommended to protect the FLASH memory against possible unwanted operation). • Half page write is possible only from SRAM. • If there are more than 32 words to write, after 32 words another Half Page programming operation starts and has to be finished. • A half page is written to the program memory only if the first address to load is the start address of a half page (multiple of 128 bytes) and the 31 remaining words to load are in the same half page. • During the Program memory half page write all read operations are forbidden (this includes DMA read operations and debugger read operations such as breakpoints, periodic updates, etc.). • If a PGERR is set during a Program memory half page write, the complete write operation is aborted. Software should then reset the FPRG and PROG/DATA bits and restart the write operation from the beginning.

HAL_FLASHEx_HalfPageProgram

Function name	<code>__RAM_FUNC HAL_FLASHEx_HalfPageProgram (uint32_t Address, uint32_t * pBuffer)</code>
---------------	--

Function description	Program a half page in program memory.
Parameters	<ul style="list-style-type: none"> • Address: specifies the address to be written. • pBuffer: pointer to the buffer containing the data to be written to the half page.
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • To correctly run this function, the HAL_FLASH_Unlock() function must be called before. Call the HAL_FLASH_Lock() to disable the flash memory access (recommended to protect the FLASH memory against possible unwanted operation) • Half page write is possible only from SRAM. • If there are more than 32 words to write, after 32 words another Half Page programming operation starts and has to be finished. • A half page is written to the program memory only if the first address to load is the start address of a half page (multiple of 128 bytes) and the 31 remaining words to load are in the same half page. • During the Program memory half page write all read operations are forbidden (this includes DMA read operations and debugger read operations such as breakpoints, periodic updates, etc.). • If a PGAEERR is set during a Program memory half page write, the complete write operation is aborted. Software should then reset the FPRG and PROG/DATA bits and restart the write operation from the beginning.

HAL_FLASHEx_GetError

Function name	_RAM_FUNC HAL_FLASHEx_GetError (uint32_t * Error)
Function description	Get the specific FLASH errors flag.
Parameters	<ul style="list-style-type: none"> • Error: pointer is the error value. It can be a mixed of: <ul style="list-style-type: none"> - HAL_FLASH_ERROR_OPTVUSR FLASH Option User validity error - HAL_FLASH_ERROR_PGA FLASH Programming Alignment error flag - HAL_FLASH_ERROR_WRP FLASH Write protected error flag - HAL_FLASH_ERROR_OPTV FLASH Option valid error flag
Return values	<ul style="list-style-type: none"> • HAL: Status

HAL_FLASHEx_DATAEEPROM_EraseDoubleWord

Function name	_RAM_FUNC HAL_FLASHEx_DATAEEPROM_EraseDoubleWord (uint32_t Address)
Function description	Erase a double word in data memory.
Parameters	<ul style="list-style-type: none"> • Address: specifies the address to be erased.
Return values	<ul style="list-style-type: none"> • HAL: status

Notes

- To correctly run this function, the HAL_FLASH_EEPROM_Unlock() function must be called before. Call the HAL_FLASH_EEPROM_Lock() to he data EEPROM access and Flash program erase control register access(recommended to protect the DATA_EEPROM against possible unwanted operation).
- Data memory double word erase is possible only from SRAM.
- A double word is erased to the data memory only if the first address to load is the start address of a double word (multiple of 8 bytes).
- During the Data memory double word erase, all read operations are forbidden (this includes DMA read operations and debugger read operations such as breakpoints, periodic updates, etc.).

HAL_FLASHEx_DATAEEPROM_ProgramDoubleWord

Function name

**_RAM_FUNC
HAL_FLASHEx_DATAEEPROM_ProgramDoubleWord
(uint32_t Address, uint64_t Data)**

Function description

Write a double word in data memory without erase.

Parameters

- **Address:** specifies the address to be written.
- **Data:** specifies the data to be written.

Return values

• **HAL:** status

Notes

- To correctly run this function, the HAL_FLASH_EEPROM_Unlock() function must be called before. Call the HAL_FLASH_EEPROM_Lock() to he data EEPROM access and Flash program erase control register access(recommended to protect the DATA_EEPROM against possible unwanted operation).
- Data memory double word write is possible only from SRAM.
- A data memory double word is written to the data memory only if the first address to load is the start address of a double word (multiple of double word).
- During the Data memory double word write, all read operations are forbidden (this includes DMA read operations and debugger read operations such as breakpoints, periodic updates, etc.).

20 HAL GPIO Generic Driver

20.1 GPIO Firmware driver registers structures

20.1.1 GPIO_InitTypeDef

Data Fields

- *uint32_t Pin*
- *uint32_t Mode*
- *uint32_t Pull*
- *uint32_t Speed*
- *uint32_t Alternate*

Field Documentation

- ***uint32_t GPIO_InitTypeDef::Pin***
Specifies the GPIO pins to be configured. This parameter can be any value of [**GPIO_pins**](#)
- ***uint32_t GPIO_InitTypeDef::Mode***
Specifies the operating mode for the selected pins. This parameter can be a value of [**GPIO_mode**](#)
- ***uint32_t GPIO_InitTypeDef::Pull***
Specifies the Pull-up or Pull-Down activation for the selected pins. This parameter can be a value of [**GPIO_pull**](#)
- ***uint32_t GPIO_InitTypeDef::Speed***
Specifies the speed for the selected pins. This parameter can be a value of [**GPIO_speed**](#)
- ***uint32_t GPIO_InitTypeDef::Alternate***
Peripheral to be connected to the selected pins This parameter can be a value of [**GPIOEx_Alternate_function_selection**](#)

20.2 GPIO Firmware driver API description

20.2.1 GPIO Peripheral features

Each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input mode
- Analog mode
- Output mode
- Alternate function mode
- External interrupt/event lines

During and just after reset, the alternate functions and external interrupt lines are not active and the I/O ports are configured in input floating mode.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not.

In Output or Alternate mode, each IO can be configured on open-drain or push-pull type and the IO speed can be selected depending on the VDD value.

The microcontroller IO pins are connected to onboard peripherals/modules through a multiplexer that allows only one peripheral's alternate function (AF) connected to an IO pin

at a time. In this way, there can be no conflict between peripherals sharing the same IO pin.

All ports have external interrupt/event capability. To use external interrupt lines, the port must be configured in input mode. All available GPIO pins are connected to the 16 external interrupt/event lines from EXTI0 to EXTI15.

The external interrupt/event controller consists of up to 28 edge detectors (depending on products 16 lines are connected to GPIO) for generating event/interrupt requests (each input line can be independently configured to select the type (interrupt or event) and the corresponding trigger event (rising or falling or both). Each line can also be masked independently.

20.2.2 How to use this driver

1. Enable the GPIO AHB clock using the following function : __GPIOx_CLK_ENABLE().
2. Configure the GPIO pin(s) using HAL_GPIO_Init().
 - Configure the IO mode using "Mode" member from GPIO_InitTypeDef structure
 - Activate Pull-up, Pull-down resistor using "Pull" member from GPIO_InitTypeDef structure.
 - In case of Output or alternate function mode selection: the speed is configured through "Speed" member from GPIO_InitTypeDef structure, the speed is configurable: Low, Medium and High.
 - If alternate mode is selected, the alternate function connected to the IO is configured through "Alternate" member from GPIO_InitTypeDef structure
 - Analog mode is required when a pin is to be used as ADC channel or DAC output.
 - In case of external interrupt/event selection the "Mode" member from GPIO_InitTypeDef structure select the type (interrupt or event) and the corresponding trigger event (rising or falling or both).
3. In case of external interrupt/event mode selection, configure NVIC IRQ priority mapped to the EXTI line using HAL_NVIC_SetPriority() and enable it using HAL_NVIC_EnableIRQ().
4. HAL_GPIO_DeInit allows to set register values to their reset value. It's also recommended to use it to unconfigure pin which was used as an external interrupt or in event mode. That's the only way to reset corresponding bit in EXTI & SYSCFG registers.
5. To get the level of a pin configured in input mode use HAL_GPIO_ReadPin().
6. To set/reset the level of a pin configured in output mode use HAL_GPIO_WritePin()/HAL_GPIO_TogglePin().
7. To lock pin configuration until next reset use HAL_GPIO_LockPin().
8. During and just after reset, the alternate functions are not active and the GPIO pins are configured in input floating mode (except JTAG pins).
9. The LSE oscillator pins OSC32_IN and OSC32_OUT can be used as general purpose (PC14 and PC15, respectively) when the LSE oscillator is off. The LSE has priority over the GPIO function.
10. The HSE oscillator pins OSC_IN/OSC_OUT can be used as general purpose PH0 and PH1, respectively, when the HSE oscillator is off. The HSE has priority over the GPIO function.

20.2.3 Initialization and Configuration functions

This section contains the following APIs:

- [**HAL_GPIO_Init\(\)**](#)
- [**HAL_GPIO_DeInit\(\)**](#)

20.2.4 Detailed description of functions

HAL_GPIO_Init

Function name **void HAL_GPIO_Init (GPIO_TypeDef * GPIOx,
GPIO_InitTypeDef * GPIO_InitStruct)**

Function description Initializes the GPIOx peripheral according to the specified parameters in the GPIO_InitStruct.

- Parameters
- **GPIOx:** where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices
 - **GPIO_InitStruct:** pointer to a GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.

Return values • **None:**

HAL_GPIO_DeInit

Function name **void HAL_GPIO_DeInit (GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)**

Function description De-initializes the GPIOx peripheral registers to their default reset values.

- Parameters
- **GPIOx:** where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices
 - **GPIO_Pin:** specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15).

Return values • **None:**

HAL_GPIO_ReadPin

Function name **GPIO_PinState HAL_GPIO_ReadPin (GPIO_TypeDef * GPIOx,
uint16_t GPIO_Pin)**

Function description Reads the specified input port pin.

- Parameters
- **GPIOx:** where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices
 - **GPIO_Pin:** specifies the port bit to read. This parameter can be GPIO_PIN_x where x can be (0..15).

Return values • **The:** input port pin value.

HAL_GPIO_WritePin

Function name **void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)**

Function description Sets or clears the selected data port bit.

- Parameters
- **GPIOx:** where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices
 - **GPIO_Pin:** specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15).
 - **PinState:** specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum

	<p>values:</p> <ul style="list-style-type: none"> - GPIO_PIN_RESET: to clear the port pin - GPIO_PIN_SET: to set the port pin
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This function uses GPIOx_BSRR register to allow atomic read/modify accesses. In this way, there is no risk of an IRQ occurring between the read and the modify access.

HAL_GPIO_TogglePin

Function name	<code>void HAL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)</code>
Function description	Toggles the specified GPIO pin.
Parameters	<ul style="list-style-type: none"> • GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices • GPIO_Pin: specifies the pins to be toggled.
Return values	<ul style="list-style-type: none"> • None:

HAL_GPIO_LockPin

Function name	<code>HAL_StatusTypeDef HAL_GPIO_LockPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)</code>
Function description	Locks GPIO Pins configuration registers.
Parameters	<ul style="list-style-type: none"> • GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices • GPIO_Pin: Specifies the port bit to be locked. This parameter can be any combination of GPIO_Pin_x where x can be (0..15).
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • The locked registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH. • The configuration of the locked GPIO pins can no longer be modified until the next reset. • Limitation concerning GPIOx_OTYPER: Locking of GPIOx_OTYPER[i] with i = 15..8 depends from setting of GPIOx_LCKR[i-8] and not from GPIOx_LCKR[i]. GPIOx_LCKR[i-8] is locking GPIOx_OTYPER[i] together with GPIOx_OTYPER[i-8]. It is not possible to lock GPIOx_OTYPER[i] with i = 15..8, without locking also GPIOx_OTYPER[i-8]. Workaround: When calling HAL_GPIO_LockPin with GPIO_Pin from GPIO_PIN_8 to GPIO_PIN_15, you must call also HAL_GPIO_LockPin with GPIO_Pin - 8. (When locking a pin from GPIO_PIN_8 to GPIO_PIN_15, you must lock also the corresponding GPIO_PIN_0 to GPIO_PIN_7).

HAL_GPIO_EXTI_IRQHandler

Function name	void HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin)
Function description	This function handles EXTI interrupt request.
Parameters	<ul style="list-style-type: none"> • GPIO_Pin: Specifies the port pin connected to corresponding EXTI line.
Return values	<ul style="list-style-type: none"> • None:

HAL_GPIO_EXTI_Callback

Function name	void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
Function description	EXTI line detection callbacks.
Parameters	<ul style="list-style-type: none"> • GPIO_Pin: Specifies the port pin connected to corresponding EXTI line.
Return values	<ul style="list-style-type: none"> • None:

20.3 GPIO Firmware driver defines

20.3.1 GPIO

GPIO Exported Macros**_HAL_GPIO_EXTI_GET_FLAG****Description:**

- Checks whether the specified EXTI line flag is set or not.

Parameters:

- **_EXTI_LINE_:** specifies the EXTI line flag to check. This parameter can be GPIO_PIN_x where x can be(0..15)

Return value:

- The: new state of **_EXTI_LINE_** (SET or RESET).

_HAL_GPIO_EXTI_CLEAR_FLAG**Description:**

- Clears the EXTI's line pending flags.

Parameters:

- **_EXTI_LINE_:** specifies the EXTI lines flags to clear. This parameter can be any combination of GPIO_PIN_x where x can be (0..15)

Return value:

- None

_HAL_GPIO_EXTI_GET_IT**Description:**

- Checks whether the specified EXTI line is asserted or not.

Parameters:

- `__EXTI_LINE__`: specifies the EXTI line to check. This parameter can be `GPIO_PIN_x` where `x` can be(0..15)

Return value:

- The: new state of `__EXTI_LINE__` (SET or RESET).

`__HAL_GPIO_EXTI_CLEAR_IT`**Description:**

- Clears the EXTI's line pending bits.

Parameters:

- `__EXTI_LINE__`: specifies the EXTI lines to clear. This parameter can be any combination of `GPIO_PIN_x` where `x` can be (0..15)

Return value:

- None

`__HAL_GPIO_EXTI_GENERATE_SWIT`**Description:**

- Generates a Software interrupt on selected EXTI line.

Parameters:

- `__EXTI_LINE__`: specifies the EXTI line to check. This parameter can be `GPIO_PIN_x` where `x` can be(0..15)

Return value:

- None

GPIO mode

<code>GPIO_MODE_INPUT</code>	Input Floating Mode
<code>GPIO_MODE_OUTPUT_PP</code>	Output Push Pull Mode
<code>GPIO_MODE_OUTPUT_OD</code>	Output Open Drain Mode
<code>GPIO_MODE_AF_PP</code>	Alternate Function Push Pull Mode
<code>GPIO_MODE_AF_OD</code>	Alternate Function Open Drain Mode
<code>GPIO_MODE_ANALOG</code>	Analog Mode
<code>GPIO_MODE_IT_RISING</code>	External Interrupt Mode with Rising edge trigger detection
<code>GPIO_MODE_IT_FALLING</code>	External Interrupt Mode with Falling edge trigger detection
<code>GPIO_MODE_IT_RISING_FALLING</code>	External Interrupt Mode with Rising/Falling edge trigger detection
<code>GPIO_MODE_EVT_RISING</code>	External Event Mode with Rising edge trigger detection

GPIO_MODE_EVT_FALLING External Event Mode with Falling edge trigger detection

GPIO_MODE_EVT_RISING_FALLING External Event Mode with Rising/Falling edge trigger detection

GPIO pins

GPIO_PIN_0

GPIO_PIN_1

GPIO_PIN_2

GPIO_PIN_3

GPIO_PIN_4

GPIO_PIN_5

GPIO_PIN_6

GPIO_PIN_7

GPIO_PIN_8

GPIO_PIN_9

GPIO_PIN_10

GPIO_PIN_11

GPIO_PIN_12

GPIO_PIN_13

GPIO_PIN_14

GPIO_PIN_15

GPIO_PIN_All

GPIO_PIN_MASK

GPIO pull

GPIO_NOPULL No Pull-up or Pull-down activation

GPIO_PULLUP Pull-up activation

GPIO_PULLDOWN Pull-down activation

GPIO speed

GPIO_SPEED_FREQ_LOW max: 400 KHz, please refer to the product datasheet

GPIO_SPEED_FREQ_MEDIUM max: 1 MHz to 2 MHz, please refer to the product datasheet

GPIO_SPEED_FREQ_HIGH max: 2 MHz to 10 MHz, please refer to the product datasheet

GPIO_SPEED_FREQ VERY_HIGH max: 8 MHz to 50 MHz, please refer to the product datasheet

21 HAL GPIO Extension Driver

21.1 GPIOEx Firmware driver defines

21.1.1 GPIOEx

GPIOEx Alternate function selection

GPIO_AF0_MCO	MCO Alternate Function mapping
GPIO_AF0_TAMPER	TAMPER Alternate Function mapping
GPIO_AF0_SWJ	SWJ (SWD and JTAG) Alternate Function mapping
GPIO_AF0_TRACE	TRACE Alternate Function mapping
GPIO_AF0_RTC_50Hz	RTC_OUT Alternate Function mapping
GPIO_AF1_TIM2	TIM2 Alternate Function mapping
GPIO_AF2_TIM3	TIM3 Alternate Function mapping
GPIO_AF2_TIM4	TIM4 Alternate Function mapping
GPIO_AF2_TIM5	TIM5 Alternate Function mapping
GPIO_AF3_TIM9	TIM9 Alternate Function mapping
GPIO_AF3_TIM10	TIM10 Alternate Function mapping
GPIO_AF3_TIM11	TIM11 Alternate Function mapping
GPIO_AF4_I2C1	I2C1 Alternate Function mapping
GPIO_AF4_I2C2	I2C2 Alternate Function mapping
GPIO_AF5_SPI1	SPI1/I2S1 Alternate Function mapping
GPIO_AF5_SPI2	SPI2/I2S2 Alternate Function mapping
GPIO_AF6_SPI3	SPI3/I2S3 Alternate Function mapping
GPIO_AF7_USART1	USART1 Alternate Function mapping
GPIO_AF7_USART2	USART2 Alternate Function mapping
GPIO_AF7_USART3	USART3 Alternate Function mapping
GPIO_AF8_UART4	UART4 Alternate Function mapping
GPIO_AF8_UART5	UART5 Alternate Function mapping
GPIO_AF11_LCD	LCD Alternate Function mapping
GPIO_AF12_FSMC	FSMC Alternate Function mapping
GPIO_AF12_SDIO	SDIO Alternate Function mapping
GPIO_AF14_TIM_IC1	TIMER INPUT CAPTURE Alternate Function mapping
GPIO_AF14_TIM_IC2	TIMER INPUT CAPTURE Alternate Function mapping
GPIO_AF14_TIM_IC3	TIMER INPUT CAPTURE Alternate Function mapping
GPIO_AF14_TIM_IC4	TIMER INPUT CAPTURE Alternate Function mapping
GPIO_AF15_EVENTOUT	EVENTOUT Alternate Function mapping

22 HAL I2C Generic Driver

22.1 I2C Firmware driver registers structures

22.1.1 I2C_InitTypeDef

Data Fields

- *uint32_t ClockSpeed*
- *uint32_t DutyCycle*
- *uint32_t OwnAddress1*
- *uint32_t AddressingMode*
- *uint32_t DualAddressMode*
- *uint32_t OwnAddress2*
- *uint32_t GeneralCallMode*
- *uint32_t NoStretchMode*

Field Documentation

- ***uint32_t I2C_InitTypeDef::ClockSpeed***
Specifies the clock frequency. This parameter must be set to a value lower than 400kHz
- ***uint32_t I2C_InitTypeDef::DutyCycle***
Specifies the I2C fast mode duty cycle. This parameter can be a value of [*I2C_duty_cycle_in_fast_mode*](#)
- ***uint32_t I2C_InitTypeDef::OwnAddress1***
Specifies the first device own address. This parameter can be a 7-bit or 10-bit address.
- ***uint32_t I2C_InitTypeDef::AddressingMode***
Specifies if 7-bit or 10-bit addressing mode is selected. This parameter can be a value of [*I2C_addressing_mode*](#)
- ***uint32_t I2C_InitTypeDef::DualAddressMode***
Specifies if dual addressing mode is selected. This parameter can be a value of [*I2C_dual_addressing_mode*](#)
- ***uint32_t I2C_InitTypeDef::OwnAddress2***
Specifies the second device own address if dual addressing mode is selected. This parameter can be a 7-bit address.
- ***uint32_t I2C_InitTypeDef::GeneralCallMode***
Specifies if general call mode is selected. This parameter can be a value of [*I2C_general_call_addressing_mode*](#)
- ***uint32_t I2C_InitTypeDef::NoStretchMode***
Specifies if nostretch mode is selected. This parameter can be a value of [*I2C_nostretch_mode*](#)

22.1.2 I2C_HandleTypeDef

Data Fields

- *I2C_TypeDef * Instance*
- *I2C_InitTypeDef Init*
- *uint8_t * pBuffPtr*
- *uint16_t XferSize*
- *__IO uint16_t XferCount*
- *__IO uint32_t XferOptions*

- `__IO uint32_t PreviousState`
- `DMA_HandleTypeDef * hdmatx`
- `DMA_HandleTypeDef * hdmarx`
- `HAL_LockTypeDef Lock`
- `__IO HAL_I2C_StateTypeDef State`
- `__IO HAL_I2C_ModeTypeDef Mode`
- `__IO uint32_t ErrorCode`
- `__IO uint32_t Devaddress`
- `__IO uint32_t Memaddress`
- `__IO uint32_t MemaddSize`
- `__IO uint32_t EventCount`

Field Documentation

- `I2C_HandleTypeDef* I2C_HandleTypeDef::Instance`
I2C registers base address
- `I2C_InitTypeDef I2C_HandleTypeDef::Init`
I2C communication parameters
- `uint8_t* I2C_HandleTypeDef::pBuffPtr`
Pointer to I2C transfer buffer
- `uint16_t I2C_HandleTypeDef::XferSize`
I2C transfer size
- `__IO uint16_t I2C_HandleTypeDef::XferCount`
I2C transfer counter
- `__IO uint32_t I2C_HandleTypeDef::XferOptions`
I2C transfer options
- `__IO uint32_t I2C_HandleTypeDef::PreviousState`
I2C communication Previous state and mode context for internal usage
- `DMA_HandleTypeDef* I2C_HandleTypeDef::hdmatx`
I2C Tx DMA handle parameters
- `DMA_HandleTypeDef* I2C_HandleTypeDef::hdmarx`
I2C Rx DMA handle parameters
- `HAL_LockTypeDef I2C_HandleTypeDef::Lock`
I2C locking object
- `__IO HAL_I2C_StateTypeDef I2C_HandleTypeDef::State`
I2C communication state
- `__IO HAL_I2C_ModeTypeDef I2C_HandleTypeDef::Mode`
I2C communication mode
- `__IO uint32_t I2C_HandleTypeDef::ErrorCode`
I2C Error code
- `__IO uint32_t I2C_HandleTypeDef::Devaddress`
I2C Target device address
- `__IO uint32_t I2C_HandleTypeDef::Memaddress`
I2C Target memory address
- `__IO uint32_t I2C_HandleTypeDef::MemaddSize`
I2C Target memory address size
- `__IO uint32_t I2C_HandleTypeDef::EventCount`
I2C Event counter

22.2 I2C Firmware driver API description

22.2.1 How to use this driver

The I2C HAL driver can be used as follows:

1. Declare a I2C_HandleTypeDef handle structure, for example: I2C_HandleTypeDef hi2c;
2. Initialize the I2C low level resources by implementing the HAL_I2C_MspInit() API:
 - a. Enable the I2Cx interface clock
 - b. I2C pins configuration
 - Enable the clock for the I2C GPIOs
 - Configure I2C pins as alternate function open-drain
 - c. NVIC configuration if you need to use interrupt process
 - Configure the I2Cx interrupt priority
 - Enable the NVIC I2C IRQ Channel
 - d. DMA Configuration if you need to use DMA process
 - Declare a DMA_HandleTypeDef handle structure for the transmit or receive channel
 - Enable the DMAx interface clock using
 - Configure the DMA handle parameters
 - Configure the DMA Tx or Rx channel
 - Associate the initialized DMA handle to the hi2c DMA Tx or Rx handle
 - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx or Rx channel
3. Configure the Communication Speed, Duty cycle, Addressing mode, Own Address1, Dual Addressing mode, Own Address2, General call and Nostretch mode in the hi2c Init structure.
4. Initialize the I2C registers by calling the HAL_I2C_Init(), configures also the low level Hardware (GPIO, CLOCK, NVIC...etc) by calling the customized HAL_I2C_MspInit(&hi2c) API.
5. To check if target device is ready for communication, use the function HAL_I2C_IsDeviceReady()
6. For I2C IO and IO MEM operations, three operation modes are available within this driver :

Polling mode IO operation

- Transmit in master mode an amount of data in blocking mode using HAL_I2C_Master_Transmit()
- Receive in master mode an amount of data in blocking mode using HAL_I2C_Master_Receive()
- Transmit in slave mode an amount of data in blocking mode using HAL_I2C_Slave_Transmit()
- Receive in slave mode an amount of data in blocking mode using HAL_I2C_Slave_Receive()

Polling mode IO MEM operation

- Write an amount of data in blocking mode to a specific memory address using HAL_I2C_Mem_Write()
- Read an amount of data in blocking mode from a specific memory address using HAL_I2C_Mem_Read()

Interrupt mode IO operation

- Transmit in master mode an amount of data in non-blocking mode using HAL_I2C_Master_Transmit_IT()
- At transmission end of transfer, HAL_I2C_MasterTxCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_MasterTxCpltCallback()

- Receive in master mode an amount of data in non-blocking mode using `HAL_I2C_Master_Receive_IT()`
- At reception end of transfer, `HAL_I2C_MasterRxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_MasterRxCpltCallback()`
- Transmit in slave mode an amount of data in non-blocking mode using `HAL_I2C_Slave_Transmit_IT()`
- At transmission end of transfer, `HAL_I2C_SlaveTxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_SlaveTxCpltCallback()`
- Receive in slave mode an amount of data in non-blocking mode using `HAL_I2C_Slave_Receive_IT()`
- At reception end of transfer, `HAL_I2C_SlaveRxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_SlaveRxCpltCallback()`
- In case of transfer Error, `HAL_I2C_ErrorCallback()` function is executed and user can add his own code by customization of function pointer `HAL_I2C_ErrorCallback()`
- Abort a master I2C process communication with Interrupt using `HAL_I2C_Master_Abort_IT()`
- End of abort process, `HAL_I2C_AbortCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_AbortCpltCallback()`
- Discard a slave I2C process communication using `_HAL_I2C_GENERATE_NACK()` macro. This action will inform Master to generate a Stop condition to discard the communication.

Interrupt mode IO sequential operation



These interfaces allow to manage a sequential transfer with a repeated start condition when a direction change during transfer

- A specific option field manage the different steps of a sequential transfer
- Option field values are defined through @ref I2C_XFEROPTIONS and are listed below:
 - `I2C_FIRST_AND_LAST_FRAME`: No sequential usage, functionnal is same as associated interfaces in no sequential mode
 - `I2C_FIRST_FRAME`: Sequential usage, this option allow to manage a sequence with start condition, address and data to transfer without a final stop condition
 - `I2C_NEXT_FRAME`: Sequential usage, this option allow to manage a sequence with a restart condition, address and with new data to transfer if the direction change or manage only the new data to transfer if no direction change and without a final stop condition in both cases
 - `I2C_LAST_FRAME`: Sequential usage, this option allow to manage a sequence with a restart condition, address and with new data to transfer if the direction change or manage only the new data to transfer if no direction change and with a final stop condition in both cases
- Differents sequential I2C interfaces are listed below:
 - Sequential transmit in master I2C mode an amount of data in non-blocking mode using `HAL_I2C_Master_Sequential_Transmit_IT()`
 - At transmission end of current frame transfer, `HAL_I2C_MasterTxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_MasterTxCpltCallback()`

- Sequential receive in master I2C mode an amount of data in non-blocking mode using HAL_I2C_Master_Sequential_Receive_IT()
 - At reception end of current frame transfer, HAL_I2C_MasterRxCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_MasterRxCpltCallback()
- Abort a master I2C process communication with Interrupt using HAL_I2C_Master_Abort_IT()
 - End of abort process, HAL_I2C_AbortCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_AbortCpltCallback()
- Enable/disable the Address listen mode in slave I2C mode using HAL_I2C_EnableListen_IT() HAL_I2C_DisableListen_IT()
 - When address slave I2C match, HAL_I2C_AddrCallback() is executed and user can add his own code to check the Address Match Code and the transmission direction request by master (Write/Read).
 - At Listen mode end HAL_I2C_ListenCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_ListenCpltCallback()
- Sequential transmit in slave I2C mode an amount of data in non-blocking mode using HAL_I2C_Slave_Sequential_Transmit_IT()
 - At transmission end of current frame transfer, HAL_I2C_SlaveTxCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_SlaveTxCpltCallback()
- Sequential receive in slave I2C mode an amount of data in non-blocking mode using HAL_I2C_Slave_Sequential_Receive_IT()
 - At reception end of current frame transfer, HAL_I2C_SlaveRxCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_SlaveRxCpltCallback()
- In case of transfer Error, HAL_I2C_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_I2C_ErrorCallback()
- Abort a master I2C process communication with Interrupt using HAL_I2C_Master_Abort_IT()
- End of abort process, HAL_I2C_AbortCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_AbortCpltCallback()
- Discard a slave I2C process communication using __HAL_I2C_GENERATE_NACK() macro. This action will inform Master to generate a Stop condition to discard the communication.

Interrupt mode IO MEM operation

- Write an amount of data in non-blocking mode with Interrupt to a specific memory address using HAL_I2C_Mem_Write_IT()
- At Memory end of write transfer, HAL_I2C_MemTxCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_MemTxCpltCallback()
- Read an amount of data in non-blocking mode with Interrupt from a specific memory address using HAL_I2C_Mem_Read_IT()
- At Memory end of read transfer, HAL_I2C_MemRxCpltCallback() is executed and user can add his own code by customization of function pointer HAL_I2C_MemRxCpltCallback()
- In case of transfer Error, HAL_I2C_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_I2C_ErrorCallback()

DMA mode IO operation

- Transmit in master mode an amount of data in non-blocking mode (DMA) using `HAL_I2C_Master_Transmit_DMA()`
- At transmission end of transfer, `HAL_I2C_MasterTxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_MasterTxCpltCallback()`
- Receive in master mode an amount of data in non-blocking mode (DMA) using `HAL_I2C_Master_Receive_DMA()`
- At reception end of transfer, `HAL_I2C_MasterRxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_MasterRxCpltCallback()`
- Transmit in slave mode an amount of data in non-blocking mode (DMA) using `HAL_I2C_Slave_Transmit_DMA()`
- At transmission end of transfer, `HAL_I2C_SlaveTxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_SlaveTxCpltCallback()`
- Receive in slave mode an amount of data in non-blocking mode (DMA) using `HAL_I2C_Slave_Receive_DMA()`
- At reception end of transfer, `HAL_I2C_SlaveRxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_SlaveRxCpltCallback()`
- In case of transfer Error, `HAL_I2C_ErrorCallback()` function is executed and user can add his own code by customization of function pointer `HAL_I2C_ErrorCallback()`
- Abort a master I2C process communication with Interrupt using `HAL_I2C_Master_Abort_IT()`
- End of abort process, `HAL_I2C_AbortCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_AbortCpltCallback()`
- Discard a slave I2C process communication using `_HAL_I2C_GENERATE_NACK()` macro. This action will inform Master to generate a Stop condition to discard the communication.

DMA mode IO MEM operation

- Write an amount of data in non-blocking mode with DMA to a specific memory address using `HAL_I2C_Mem_Write_DMA()`
- At Memory end of write transfer, `HAL_I2C_MemTxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_MemTxCpltCallback()`
- Read an amount of data in non-blocking mode with DMA from a specific memory address using `HAL_I2C_Mem_Read_DMA()`
- At Memory end of read transfer, `HAL_I2C_MemRxCpltCallback()` is executed and user can add his own code by customization of function pointer `HAL_I2C_MemRxCpltCallback()`
- In case of transfer Error, `HAL_I2C_ErrorCallback()` function is executed and user can add his own code by customization of function pointer `HAL_I2C_ErrorCallback()`

I2C HAL driver macros list

Below the list of most used macros in I2C HAL driver.

- `_HAL_I2C_ENABLE`: Enable the I2C peripheral
- `_HAL_I2C_DISABLE`: Disable the I2C peripheral
- `_HAL_I2C_GENERATE_NACK`: Generate a Non-Acknowledge I2C peripheral in Slave mode
- `_HAL_I2C_GET_FLAG`: Check whether the specified I2C flag is set or not

- `_HAL_I2C_CLEAR_FLAG`: Clear the specified I2C pending flag
- `_HAL_I2C_ENABLE_IT`: Enable the specified I2C interrupt
- `_HAL_I2C_DISABLE_IT`: Disable the specified I2C interrupt



You can refer to the I2C HAL driver header file for more useful macros

22.2.2 Initialization and de-initialization functions

This subsection provides a set of functions allowing to initialize and deinitialize the I2Cx peripheral:

- User must Implement `HAL_I2C_MspInit()` function in which he configures all related peripherals resources (CLOCK, GPIO, DMA, IT and NVIC).
- Call the function `HAL_I2C_Init()` to configure the selected device with the selected configuration:
 - Communication Speed
 - Duty cycle
 - Addressing mode
 - Own Address 1
 - Dual Addressing mode
 - Own Address 2
 - General call mode
 - Nostretch mode
- Call the function `HAL_I2C_DeInit()` to restore the default configuration of the selected I2Cx peripheral.

This section contains the following APIs:

- `HAL_I2C_Init\(\)`
- `HAL_I2C_DeInit\(\)`
- `HAL_I2C_MspInit\(\)`
- `HAL_I2C_MspDeInit\(\)`

22.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the I2C data transfers.

1. There are two modes of transfer:
 - Blocking mode : The communication is performed in the polling mode. The status of all data processing is returned by the same function after finishing transfer.
 - No-Blocking mode : The communication is performed using Interrupts or DMA. These functions return the status of the transfer startup. The end of the data processing will be indicated through the dedicated I2C IRQ when using Interrupt mode or the DMA IRQ when using DMA mode.
2. Blocking mode functions are :
 - `HAL_I2C_Master_Transmit()`
 - `HAL_I2C_Master_Receive()`
 - `HAL_I2C_Slave_Transmit()`
 - `HAL_I2C_Slave_Receive()`
 - `HAL_I2C_Mem_Write()`
 - `HAL_I2C_Mem_Read()`
 - `HAL_I2C_IsDeviceReady()`
3. No-Blocking mode functions with Interrupt are :

- HAL_I2C_Master_Transmit_IT()
 - HAL_I2C_Master_Receive_IT()
 - HAL_I2C_Slave_Transmit_IT()
 - HAL_I2C_Slave_Receive_IT()
 - HAL_I2C_Master_Sequential_Transmit_IT()
 - HAL_I2C_Master_Sequential_Receive_IT()
 - HAL_I2C_Slave_Sequential_Transmit_IT()
 - HAL_I2C_Slave_Sequential_Receive_IT()
 - HAL_I2C_Mem_Write_IT()
 - HAL_I2C_Mem_Read_IT()
4. No-Blocking mode functions with DMA are :
- HAL_I2C_Master_Transmit_DMA()
 - HAL_I2C_Master_Receive_DMA()
 - HAL_I2C_Slave_Transmit_DMA()
 - HAL_I2C_Slave_Receive_DMA()
 - HAL_I2C_Mem_Write_DMA()
 - HAL_I2C_Mem_Read_DMA()
5. A set of Transfer Complete Callbacks are provided in non Blocking mode:
- HAL_I2C_MemTxCpltCallback()
 - HAL_I2C_MemRxCpltCallback()
 - HAL_I2C_MasterTxCpltCallback()
 - HAL_I2C_MasterRxCpltCallback()
 - HAL_I2C_SlaveTxCpltCallback()
 - HAL_I2C_SlaveRxCpltCallback()
 - HAL_I2C_ErrorCallback()
 - HAL_I2C_AbortCpltCallback()

This section contains the following APIs:

- [**HAL_I2C_Master_Transmit\(\)**](#)
- [**HAL_I2C_Master_Receive\(\)**](#)
- [**HAL_I2C_Slave_Transmit\(\)**](#)
- [**HAL_I2C_Slave_Receive\(\)**](#)
- [**HAL_I2C_Master_Transmit_IT\(\)**](#)
- [**HAL_I2C_Master_Receive_IT\(\)**](#)
- [**HAL_I2C_Master_Sequential_Transmit_IT\(\)**](#)
- [**HAL_I2C_Master_Sequential_Receive_IT\(\)**](#)
- [**HAL_I2C_Slave_Transmit_IT\(\)**](#)
- [**HAL_I2C_Slave_Receive_IT\(\)**](#)
- [**HAL_I2C_Slave_Sequential_Transmit_IT\(\)**](#)
- [**HAL_I2C_Slave_Sequential_Receive_IT\(\)**](#)
- [**HAL_I2C_EnableListen_IT\(\)**](#)
- [**HAL_I2C_DisableListen_IT\(\)**](#)
- [**HAL_I2C_Master_Transmit_DMA\(\)**](#)
- [**HAL_I2C_Master_Receive_DMA\(\)**](#)
- [**HAL_I2C_Master_Abort_IT\(\)**](#)
- [**HAL_I2C_Slave_Transmit_DMA\(\)**](#)
- [**HAL_I2C_Slave_Receive_DMA\(\)**](#)
- [**HAL_I2C_Mem_Write\(\)**](#)
- [**HAL_I2C_Mem_Read\(\)**](#)
- [**HAL_I2C_Mem_Write_IT\(\)**](#)
- [**HAL_I2C_Mem_Read_IT\(\)**](#)
- [**HAL_I2C_Mem_Write_DMA\(\)**](#)
- [**HAL_I2C_Mem_Read_DMA\(\)**](#)

- `HAL_I2C_IsDeviceReady()`
- `HAL_I2C_EV_IRQHandler()`
- `HAL_I2C_ER_IRQHandler()`
- `HAL_I2C_MasterTxCpltCallback()`
- `HAL_I2C_MasterRxCpltCallback()`
- `HAL_I2C_SlaveTxCpltCallback()`
- `HAL_I2C_SlaveRxCpltCallback()`
- `HAL_I2C_AddrCallback()`
- `HAL_I2C_ListenCpltCallback()`
- `HAL_I2C_MemTxCpltCallback()`
- `HAL_I2C_MemRxCpltCallback()`
- `HAL_I2C_ErrorCallback()`
- `HAL_I2C_AbortCpltCallback()`

22.2.4 Peripheral State, Mode and Error functions

This subsection permits to get in run-time the status of the peripheral and the data flow.

This section contains the following APIs:

- `HAL_I2C_GetState()`
- `HAL_I2C_GetMode()`
- `HAL_I2C_GetError()`

22.2.5 Detailed description of functions

`HAL_I2C_Init`

Function name	<code>HAL_StatusTypeDef HAL_I2C_Init (I2C_HandleTypeDef * hi2c)</code>
Function description	Initializes the I2C according to the specified parameters in the I2C_InitTypeDef and initialize the associated handle.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • HAL: status

`HAL_I2C_DelInit`

Function name	<code>HAL_StatusTypeDef HAL_I2C_DelInit (I2C_HandleTypeDef * hi2c)</code>
Function description	Deinitialize the I2C peripheral.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • HAL: status

`HAL_I2C_MspInit`

Function name	<code>void HAL_I2C_MspInit (I2C_HandleTypeDef * hi2c)</code>
Function description	Initialize the I2C MSP.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

Return values	<ul style="list-style-type: none"> None:
---------------	---

HAL_I2C_MspDelInit

Function name	void HAL_I2C_MspDelInit (I2C_HandleTypeDef * hi2c)
Function description	DeInitialize the I2C MSP.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> None:

HAL_I2C_Master_Transmit

Function name	HAL_StatusTypeDef HAL_I2C_Master_Transmit (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Transmits in master mode an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface pData: Pointer to data buffer Size: Amount of data to be sent Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status

HAL_I2C_Master_Receive

Function name	HAL_StatusTypeDef HAL_I2C_Master_Receive (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Receives in master mode an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface pData: Pointer to data buffer Size: Amount of data to be sent Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status

HAL_I2C_Slave_Transmit

Function name	HAL_StatusTypeDef HAL_I2C_Slave_Transmit (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Transmits in slave mode an amount of data in blocking mode.

Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. pData: Pointer to data buffer Size: Amount of data to be sent Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status

HAL_I2C_Slave_Receive

Function name	HAL_StatusTypeDef HAL_I2C_Slave_Receive (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Receive in slave mode an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. pData: Pointer to data buffer Size: Amount of data to be sent Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status

HAL_I2C_Mem_Write

Function name	HAL_StatusTypeDef HAL_I2C_Mem_Write (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Write an amount of data in blocking mode to a specific memory address.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface MemAddress: Internal memory address MemAddSize: Size of internal memory address pData: Pointer to data buffer Size: Amount of data to be sent Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status

HAL_I2C_Mem_Read

Function name	HAL_StatusTypeDef HAL_I2C_Mem_Read (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Read an amount of data in blocking mode from a specific memory address.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains

- the configuration information for the specified I2C.
 - **DevAddress:** Target device address: The device 7 bits address value in datasheet must be shift at right before call interface
 - **MemAddress:** Internal memory address
 - **MemAddSize:** Size of internal memory address
 - **pData:** Pointer to data buffer
 - **Size:** Amount of data to be sent
 - **Timeout:** Timeout duration
- Return values
- **HAL:** status

HAL_I2C_IsDeviceReady

Function name	HAL_StatusTypeDef HAL_I2C_IsDeviceReady (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint32_t Trials, uint32_t Timeout)
Function description	Checks if target device is ready for communication.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • Trials: Number of trials • Timeout: Timeout duration
Return values	• HAL: status
Notes	• This function is used with Memory devices

HAL_I2C_Master_Transmit_IT

Function name	HAL_StatusTypeDef HAL_I2C_Master_Transmit_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)
Function description	Transmit in master mode an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	• HAL: status

HAL_I2C_Master_Receive_IT

Function name	HAL_StatusTypeDef HAL_I2C_Master_Receive_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)
Function description	Receive in master mode an amount of data in non-blocking mode

	with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_Slave_Transmit_IT

Function name	HAL_StatusTypeDef HAL_I2C_Slave_Transmit_IT (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)
Function description	Transmit in slave mode an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_Slave_Receive_IT

Function name	HAL_StatusTypeDef HAL_I2C_Slave_Receive_IT (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)
Function description	Receive in slave mode an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_Mem_Write_IT

Function name	HAL_StatusTypeDef HAL_I2C_Mem_Write_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size)
Function description	Write an amount of data in non-blocking mode with Interrupt to a specific memory address.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • MemAddress: Internal memory address • MemAddSize: Size of internal memory address

Return values	<ul style="list-style-type: none"> • pData: Pointer to data buffer • Size: Amount of data to be sent • HAL: status
---------------	--

HAL_I2C_Mem_Read_IT

Function name	HAL_StatusTypeDef HAL_I2C_Mem_Read_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size)
Function description	Read an amount of data in non-blocking mode with Interrupt from a specific memory address.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • MemAddress: Internal memory address • MemAddSize: Size of internal memory address • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	• HAL: status

HAL_I2C_Master_Sequential_Transmit_IT

Function name	HAL_StatusTypeDef HAL_I2C_Master_Sequential_Transmit_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
Function description	Sequential transmit in master mode an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • pData: Pointer to data buffer • Size: Amount of data to be sent • XferOptions: Options of Transfer, value of I2C XferOptions definition
Return values	• HAL: status
Notes	<ul style="list-style-type: none"> • This interface allow to manage repeated start condition when a direction change during transfer

HAL_I2C_Master_Sequential_Receive_IT

Function name	HAL_StatusTypeDef HAL_I2C_Master_Sequential_Receive_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
---------------	---

Function description	Sequential receive in master mode an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface pData: Pointer to data buffer Size: Amount of data to be sent XferOptions: Options of Transfer, value of I2C XferOptions definition
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> This interface allow to manage repeated start condition when a direction change during transfer

HAL_I2C_Slave_Sequential_Transmit_IT

Function name	HAL_StatusTypeDef HAL_I2C_Slave_Sequential_Transmit_IT(I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
Function description	Sequential transmit in slave mode an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. pData: Pointer to data buffer Size: Amount of data to be sent XferOptions: Options of Transfer, value of I2C XferOptions definition
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> This interface allow to manage repeated start condition when a direction change during transfer

HAL_I2C_Slave_Sequential_Receive_IT

Function name	HAL_StatusTypeDef HAL_I2C_Slave_Sequential_Receive_IT(I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
Function description	Sequential receive in slave mode an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. pData: Pointer to data buffer Size: Amount of data to be sent XferOptions: Options of Transfer, value of I2C XferOptions definition
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> This interface allow to manage repeated start condition when a direction change during transfer

HAL_I2C_Master_Abort_IT

Function name	HAL_StatusTypeDef HAL_I2C_Master_Abort_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress)
Function description	Abort a master I2C process communication with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • This abort can be called only if state is ready

HAL_I2C_EnableListen_IT

Function name	HAL_StatusTypeDef HAL_I2C_EnableListen_IT (I2C_HandleTypeDef * hi2c)
Function description	Enable the Address listen mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_DisableListen_IT

Function name	HAL_StatusTypeDef HAL_I2C_DisableListen_IT (I2C_HandleTypeDef * hi2c)
Function description	Disable the Address listen mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_Master_Transmit_DMA

Function name	HAL_StatusTypeDef HAL_I2C_Master_Transmit_DMA (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)
Function description	Transmit in master mode an amount of data in non-blocking mode with DMA.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_Master_Receive_DMA

Function name	HAL_StatusTypeDef HAL_I2C_Master_Receive_DMA (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)
Function description	Receive in master mode an amount of data in non-blocking mode with DMA.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_Slave_Transmit_DMA

Function name	HAL_StatusTypeDef HAL_I2C_Slave_Transmit_DMA (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)
Function description	Transmit in slave mode an amount of data in non-blocking mode with DMA.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_Slave_Receive_DMA

Function name	HAL_StatusTypeDef HAL_I2C_Slave_Receive_DMA (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)
Function description	Receive in slave mode an amount of data in non-blocking mode with DMA.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2C_Mem_Write_DMA

Function name	HAL_StatusTypeDef HAL_I2C_Mem_Write_DMA (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size)
Function description	Write an amount of data in non-blocking mode with DMA to a specific memory address.

Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface MemAddress: Internal memory address MemAddSize: Size of internal memory address pData: Pointer to data buffer Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> HAL: status

HAL_I2C_Mem_Read_DMA

Function name	HAL_StatusTypeDef HAL_I2C_Mem_Read_DMA(I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size)
Function description	Reads an amount of data in non-blocking mode with DMA from a specific memory address.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. DevAddress: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface MemAddress: Internal memory address MemAddSize: Size of internal memory address pData: Pointer to data buffer Size: Amount of data to be read
Return values	<ul style="list-style-type: none"> HAL: status

HAL_I2C_EV_IRQHandler

Function name	void HAL_I2C_EV_IRQHandler(I2C_HandleTypeDef * hi2c)
Function description	This function handles I2C event interrupt request.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> None:

HAL_I2C_ER_IRQHandler

Function name	void HAL_I2C_ER_IRQHandler(I2C_HandleTypeDef * hi2c)
Function description	This function handles I2C error interrupt request.
Parameters	<ul style="list-style-type: none"> hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> None:

HAL_I2C_MasterTxCpltCallback

Function name	void HAL_I2C_MasterTxCpltCallback (I2C_HandleTypeDef * hi2c)
Function description	Master Tx Transfer completed callback.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • None:

HAL_I2C_MasterRxCpltCallback

Function name	void HAL_I2C_MasterRxCpltCallback (I2C_HandleTypeDef * hi2c)
Function description	Master Rx Transfer completed callback.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • None:

HAL_I2C_SlaveTxCpltCallback

Function name	void HAL_I2C_SlaveTxCpltCallback (I2C_HandleTypeDef * hi2c)
Function description	Slave Tx Transfer completed callback.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • None:

HAL_I2C_SlaveRxCpltCallback

Function name	void HAL_I2C_SlaveRxCpltCallback (I2C_HandleTypeDef * hi2c)
Function description	Slave Rx Transfer completed callback.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • None:

HAL_I2C_AddrCallback

Function name	void HAL_I2C_AddrCallback (I2C_HandleTypeDef * hi2c, uint8_t TransferDirection, uint16_t AddrMatchCode)
Function description	Slave Address Match callback.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. • TransferDirection: Master request Transfer Direction (Write/Read), value of I2C XferDirection definition Master Point of View • AddrMatchCode: Address Match Code

Return values

- **None:**

HAL_I2C_ListenCpltCallback

Function name **void HAL_I2C_ListenCpltCallback (I2C_HandleTypeDef * hi2c)**

Function description Listen Complete callback.

Parameters

- **hi2c:** Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

Return values

- **None:**

HAL_I2C_MemTxCpltCallback

Function name **void HAL_I2C_MemTxCpltCallback (I2C_HandleTypeDef * hi2c)**

Function description Memory Tx Transfer completed callback.

Parameters

- **hi2c:** Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

Return values

- **None:**

HAL_I2C_MemRxCpltCallback

Function name **void HAL_I2C_MemRxCpltCallback (I2C_HandleTypeDef * hi2c)**

Function description Memory Rx Transfer completed callback.

Parameters

- **hi2c:** Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

Return values

- **None:**

HAL_I2C_ErrorCallback

Function name **void HAL_I2C_ErrorCallback (I2C_HandleTypeDef * hi2c)**

Function description I2C error callback.

Parameters

- **hi2c:** Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

Return values

- **None:**

HAL_I2C_AbortCpltCallback

Function name **void HAL_I2C_AbortCpltCallback (I2C_HandleTypeDef * hi2c)**

Function description I2C abort callback.

Parameters

- **hi2c:** Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.

Return values

- **None:**

HAL_I2C_GetState

Function name	HAL_I2C_StateTypeDef HAL_I2C_GetState (I2C_HandleTypeDef * hi2c)
Function description	Return the I2C handle state.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_I2C_GetMode

Function name	HAL_I2C_ModeTypeDef HAL_I2C_GetMode (I2C_HandleTypeDef * hi2c)
Function description	Return the I2C Master, Slave, Memory or no mode.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module
Return values	<ul style="list-style-type: none"> • HAL: mode

HAL_I2C_GetError

Function name	uint32_t HAL_I2C_GetError (I2C_HandleTypeDef * hi2c)
Function description	Return the I2C error code.
Parameters	<ul style="list-style-type: none"> • hi2c: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C.
Return values	<ul style="list-style-type: none"> • I2C: Error Code

22.3 I2C Firmware driver defines

22.3.1 I2C

I2C addressing mode

I2C_ADDRESSINGMODE_7BIT
I2C_ADDRESSINGMODE_10BIT

I2C dual addressing mode

I2C_DUALADDRESS_DISABLE
I2C_DUALADDRESS_ENABLE

I2C duty cycle in fast mode

I2C_DUTYCYCLE_2
I2C_DUTYCYCLE_16_9

I2C Error Code definition

HAL_I2C_ERROR_NONE	No error
HAL_I2C_ERROR_BERR	BERR error
HAL_I2C_ERROR_ARLO	ARLO error

<code>HAL_I2C_ERROR_AF</code>	AF error
<code>HAL_I2C_ERROR_OVR</code>	OVR error
<code>HAL_I2C_ERROR_DMA</code>	DMA transfer error
<code>HAL_I2C_ERROR_TIMEOUT</code>	Timeout Error

I2C Exported Macros

<code>_HAL_I2C_RESET_HANDLE_STATE</code>	Description:
	<ul style="list-style-type: none"> • Reset I2C handle state. Parameters: <ul style="list-style-type: none"> • <code>_HANDLE_</code>: specifies the I2C Handle. Return value: <ul style="list-style-type: none"> • None
<code>_HAL_I2C_ENABLE_IT</code>	Description:
	<ul style="list-style-type: none"> • Enable the specified I2C interrupt. Parameters: <ul style="list-style-type: none"> • <code>_HANDLE_</code>: specifies the I2C Handle. • <code>_INTERRUPT_</code>: specifies the interrupt source to enable. This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>I2C_IT_BUF</code>: Buffer interrupt enable - <code>I2C_IT_EVT</code>: Event interrupt enable - <code>I2C_IT_ERR</code>: Error interrupt enable Return value: <ul style="list-style-type: none"> • None
<code>_HAL_I2C_DISABLE_IT</code>	Description:
	<ul style="list-style-type: none"> • Disable the specified I2C interrupt. Parameters: <ul style="list-style-type: none"> • <code>_HANDLE_</code>: specifies the I2C Handle. • <code>_INTERRUPT_</code>: specifies the interrupt source to disable. This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>I2C_IT_BUF</code>: Buffer interrupt enable - <code>I2C_IT_EVT</code>: Event interrupt enable - <code>I2C_IT_ERR</code>: Error interrupt enable Return value: <ul style="list-style-type: none"> • None
<code>_HAL_I2C_GET_IT_SOURCE</code>	Description:
	<ul style="list-style-type: none"> • Check whether the specified I2C interrupt source is enabled or not. Parameters: <ul style="list-style-type: none"> • <code>_HANDLE_</code>: specifies the I2C Handle. • <code>_INTERRUPT_</code>: specifies the I2C

interrupt source to check. This parameter can be one of the following values:

- I2C_IT_BUF: Buffer interrupt enable
- I2C_IT_EVT: Event interrupt enable
- I2C_IT_ERR: Error interrupt enable

Return value:

- The: new state of `__INTERRUPT__` (TRUE or FALSE).

`__HAL_I2C_GET_FLAG`

Description:

- Check whether the specified I2C flag is set or not.

Parameters:

- `__HANDLE__`: specifies the I2C Handle.
- `__FLAG__`: specifies the flag to check. This parameter can be one of the following values:
 - I2C_FLAG_OVR: Overrun/Underrun flag
 - I2C_FLAG_AF: Acknowledge failure flag
 - I2C_FLAG_ARLO: Arbitration lost flag
 - I2C_FLAG_BERR: Bus error flag
 - I2C_FLAG_TXE: Data register empty flag
 - I2C_FLAG_RXNE: Data register not empty flag
 - I2C_FLAG_STOPF: Stop detection flag
 - I2C_FLAG_ADD10: 10-bit header sent flag
 - I2C_FLAG_BTF: Byte transfer finished flag
 - I2C_FLAG_ADDR: Address sent flag
Address matched flag
 - I2C_FLAG_SB: Start bit flag
 - I2C_FLAG_DUALF: Dual flag
 - I2C_FLAG_GENCALL: General call header flag
 - I2C_FLAG_TRA: Transmitter/Receiver flag
 - I2C_FLAG_BUSY: Bus busy flag
 - I2C_FLAG_MSL: Master/Slave flag

Return value:

- The: new state of `__FLAG__` (TRUE or FALSE).

`__HAL_I2C_CLEAR_FLAG`

Description:

- Clear the I2C pending flags which are cleared by writing 0 in a specific bit.

Parameters:

- `__HANDLE__`: specifies the I2C Handle.
- `__FLAG__`: specifies the flag to clear. This parameter can be any combination of the following values:
 - I2C_FLAG_OVR: Overrun/Underrun flag (Slave mode)
 - I2C_FLAG_AF: Acknowledge failure flag
 - I2C_FLAG_ARLO: Arbitration lost flag (Master mode)
 - I2C_FLAG_BERR: Bus error flag

Return value:

- None

`__HAL_I2C_CLEAR_ADDRFLAG`

Description:

- Clears the I2C ADDR pending flag.

Parameters:

- `__HANDLE__`: specifies the I2C Handle.

Return value:

- None

`__HAL_I2C_CLEAR_STOPFLAG`

Description:

- Clears the I2C STOPF pending flag.

Parameters:

- `__HANDLE__`: specifies the I2C Handle.

Return value:

- None

`__HAL_I2C_ENABLE`

Description:

- Enable the I2C peripheral.

Parameters:

- `__HANDLE__`: specifies the I2C Handle.
This parameter can be I2Cx where x: 1 or 2 to select the I2C peripheral.

Return value:

- None

`__HAL_I2C_DISABLE`

Description:

- Disable the I2C peripheral.

Parameters:

- `__HANDLE__`: specifies the I2C Handle.
This parameter can be I2Cx where x: 1 or 2 to select the I2C peripheral.

Return value:

- None

I2C Flag definition

I2C_FLAG_OVR
I2C_FLAG_AF
I2C_FLAG_ARLO
I2C_FLAG_BERR
I2C_FLAG_TXE
I2C_FLAG_RXNE
I2C_FLAG_STOPF
I2C_FLAG_ADD10
I2C_FLAG_BTF
I2C_FLAG_ADDR
I2C_FLAG_SB
I2C_FLAG_DUALF
I2C_FLAG_GENCALL
I2C_FLAG_TRA
I2C_FLAG_BUSY
I2C_FLAG_MSL

I2C general call addressing mode

I2C_GENERALCALL_DISABLE
I2C_GENERALCALL_ENABLE

I2C Interrupt configuration definition

I2C_IT_BUF
I2C_IT_EVT
I2C_IT_ERR

I2C Memory Address Size

I2C_MEMADD_SIZE_8BIT
I2C_MEMADD_SIZE_16BIT

I2C nostretch mode

I2C_NOSTRETCH_DISABLE
I2C_NOSTRETCH_ENABLE

I2C XferDirection definition Master Point of View

I2C_DIRECTION_RECEIVE
I2C_DIRECTION_TRANSMIT

I2C XferOptions definition

I2C_FIRST_FRAME

I2C_NEXT_FRAME
I2C_FIRST_AND_LAST_FRAME
I2C_LAST_FRAME

23 HAL I2S Generic Driver

23.1 I2S Firmware driver registers structures

23.1.1 I2S_InitTypeDef

Data Fields

- *uint32_t Mode*
- *uint32_t Standard*
- *uint32_t DataFormat*
- *uint32_t MCLKOutput*
- *uint32_t AudioFreq*
- *uint32_t CPOL*

Field Documentation

- ***uint32_t I2S_InitTypeDef::Mode***
Specifies the I2S operating mode. This parameter can be a value of [I2S_Mode](#)
- ***uint32_t I2S_InitTypeDef::Standard***
Specifies the standard used for the I2S communication. This parameter can be a value of [I2S_Standard](#)
- ***uint32_t I2S_InitTypeDef::DataFormat***
Specifies the data format for the I2S communication. This parameter can be a value of [I2S_Data_Format](#)
- ***uint32_t I2S_InitTypeDef::MCLKOutput***
Specifies whether the I2S MCLK output is enabled or not. This parameter can be a value of [I2S_MCLK_Output](#)
- ***uint32_t I2S_InitTypeDef::AudioFreq***
Specifies the frequency selected for the I2S communication. This parameter can be a value of [I2S_Audio_Frequency](#)
- ***uint32_t I2S_InitTypeDef::CPOL***
Specifies the idle state of the I2S clock. This parameter can be a value of [I2S_Clock_Polarity](#)

23.1.2 I2S_HandleTypeDef

Data Fields

- *SPI_TypeDef * Instance*
- *I2S_InitTypeDef Init*
- *uint16_t * pTxBuffPtr*
- *__IO uint16_t TxXferSize*
- *__IO uint16_t TxXferCount*
- *uint16_t * pRxBuffPtr*
- *__IO uint16_t RxXferSize*
- *__IO uint16_t RxXferCount*
- *DMA_HandleTypeDef * hdmatx*
- *DMA_HandleTypeDef * hdmarx*
- *__IO HAL_LockTypeDef Lock*
- *__IO HAL_I2S_StateTypeDef State*
- *__IO uint32_t ErrorCode*

Field Documentation

- `SPI_TypeDef* I2S_HandleTypeDef::Instance`
- `I2S_InitTypeDef I2S_HandleTypeDef::Init`
- `uint16_t* I2S_HandleTypeDef::pTxBuffPtr`
- `_IO uint16_t I2S_HandleTypeDef::TxXferSize`
- `_IO uint16_t I2S_HandleTypeDef::TxXferCount`
- `uint16_t* I2S_HandleTypeDef::pRxBuffPtr`
- `_IO uint16_t I2S_HandleTypeDef::RxXferSize`
- `_IO uint16_t I2S_HandleTypeDef::RxXferCount`
- `DMA_HandleTypeDef* I2S_HandleTypeDef::hdmatx`
- `DMA_HandleTypeDef* I2S_HandleTypeDef::hdmarx`
- `_IO HAL_LockTypeDef I2S_HandleTypeDef::Lock`
- `_IO HAL_I2S_StateTypeDef I2S_HandleTypeDef::State`
- `_IO uint32_t I2S_HandleTypeDef::ErrorCode`

23.2 I2S Firmware driver API description

23.2.1 How to use this driver

The I2S HAL driver can be used as follow:

1. Declare a I2S_HandleTypeDef handle structure.
2. Initialize the I2S low level resources by implement the HAL_I2S_MspInit() API:
 - a. Enable the SPIx interface clock.
 - b. I2S pins configuration:
 - Enable the clock for the I2S GPIOs.
 - Configure these I2S pins as alternate function.
 - c. NVIC configuration if you need to use interrupt process (HAL_I2S_Transmit_IT() and HAL_I2S_Receive_IT() APIs).
 - Configure the I2Sx interrupt priority.
 - Enable the NVIC I2S IRQ handle.
 - d. DMA Configuration if you need to use DMA process (HAL_I2S_Transmit_DMA() and HAL_I2S_Receive_DMA() APIs):
 - Declare a DMA handle structure for the Tx/Rx Channel.
 - Enable the DMAx interface clock.
 - Configure the declared DMA handle structure with the required Tx/Rx parameters.
 - Configure the DMA Tx/Rx Channel.
 - Associate the initialized DMA handle to the I2S DMA Tx/Rx handle.
 - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx/Rx Channel.
3. Program the Mode, Standard, Data Format, MCLK Output, Audio frequency and Polarity using HAL_I2S_Init() function. The specific I2S interrupts (Transmission complete interrupt, RXNE interrupt and Error Interrupts) will be managed using the macros `_HAL_I2S_ENABLE_IT()` and `_HAL_I2S_DISABLE_IT()` inside the transmit and receive process. Make sure that either: External clock source is configured after setting correctly the define constant HSE_VALUE in the `stm32l1xx_hal_conf.h` file.
4. Three mode of operations are available within this driver :

Polling mode IO operation

- Send an amount of data in blocking mode using `HAL_I2S_Transmit()`
- Receive an amount of data in blocking mode using `HAL_I2S_Receive()`

Interrupt mode IO operation

- Send an amount of data in non blocking mode using HAL_I2S_Transmit_IT()
- At transmission end of half transfer HAL_I2S_TxHalfCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2S_TxHalfCpltCallback
- At transmission end of transfer HAL_I2S_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2S_TxCpltCallback
- Receive an amount of data in non blocking mode using HAL_I2S_Receive_IT()
- At reception end of half transfer HAL_I2S_RxHalfCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2S_RxHalfCpltCallback
- At reception end of transfer HAL_I2S_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2S_RxCpltCallback
- In case of transfer Error, HAL_I2S_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_I2S_ErrorCallback

DMA mode IO operation

- Send an amount of data in non blocking mode (DMA) using HAL_I2S_Transmit_DMA()
- At transmission end of half transfer HAL_I2S_TxHalfCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2S_TxHalfCpltCallback
- At transmission end of transfer HAL_I2S_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2S_TxCpltCallback
- Receive an amount of data in non blocking mode (DMA) using HAL_I2S_Receive_DMA()
- At reception end of half transfer HAL_I2S_RxHalfCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2S_RxHalfCpltCallback
- At reception end of transfer HAL_I2S_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_I2S_RxCpltCallback
- In case of transfer Error, HAL_I2S_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_I2S_ErrorCallback
- Pause the DMA Transfer using HAL_I2S_DMAPause()
- Resume the DMA Transfer using HAL_I2S_DMAResume()
- Stop the DMA Transfer using HAL_I2S_DMAStop()

I2S HAL driver macros list

Below the list of most used macros in USART HAL driver.

- __HAL_I2S_ENABLE: Enable the specified SPI peripheral (in I2S mode)
- __HAL_I2S_DISABLE: Disable the specified SPI peripheral (in I2S mode)
- __HAL_I2S_ENABLE_IT : Enable the specified I2S interrupts
- __HAL_I2S_DISABLE_IT : Disable the specified I2S interrupts
- __HAL_I2S_GET_FLAG: Check whether the specified I2S flag is set or not



You can refer to the I2S HAL driver header file for more useful macros

23.2.2 Initialization and de-initialization functions

This subsection provides a set of functions allowing to initialize and de-initialize the I2Sx peripheral in simplex mode:

- User must Implement HAL_I2S_MspInit() function in which he configures all related peripherals resources (CLOCK, GPIO, DMA, IT and NVIC).
- Call the function HAL_I2S_Init() to configure the selected device with the selected configuration:
 - Mode
 - Standard
 - Data Format
 - MCLK Output
 - Audio frequency
 - Polarity
- Call the function HAL_I2S_DelInit() to restore the default configuration of the selected I2Sx peripheral.

This section contains the following APIs:

- [**HAL_I2S_Init\(\)**](#)
- [**HAL_I2S_DelInit\(\)**](#)
- [**HAL_I2S_MspInit\(\)**](#)
- [**HAL_I2S_MspDelInit\(\)**](#)

23.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the I2S data transfers.

1. There are two modes of transfer:
 - Blocking mode : The communication is performed in the polling mode. The status of all data processing is returned by the same function after finishing transfer.
 - No-Blocking mode : The communication is performed using Interrupts or DMA. These functions return the status of the transfer startup. The end of the data processing will be indicated through the dedicated I2S IRQ when using Interrupt mode or the DMA IRQ when using DMA mode.
2. Blocking mode functions are :
 - HAL_I2S_Transmit()
 - HAL_I2S_Receive()
3. No-Blocking mode functions with Interrupt are :
 - HAL_I2S_Transmit_IT()
 - HAL_I2S_Receive_IT()
4. No-Blocking mode functions with DMA are :
 - HAL_I2S_Transmit_DMA()
 - HAL_I2S_Receive_DMA()
5. A set of Transfer Complete Callbacks are provided in non Blocking mode:
 - HAL_I2S_TxCpltCallback()
 - HAL_I2S_RxCpltCallback()
 - HAL_I2S_ErrorCallback()

This section contains the following APIs:

- [**HAL_I2S_Transmit\(\)**](#)
- [**HAL_I2S_Receive\(\)**](#)
- [**HAL_I2S_Transmit_IT\(\)**](#)
- [**HAL_I2S_Receive_IT\(\)**](#)
- [**HAL_I2S_Transmit_DMA\(\)**](#)

- [*HAL_I2S_Receive_DMA\(\)*](#)
- [*HAL_I2S_DMAPause\(\)*](#)
- [*HAL_I2S_DMAResume\(\)*](#)
- [*HAL_I2S_DMAStop\(\)*](#)
- [*HAL_I2S_IRQHandler\(\)*](#)
- [*HAL_I2S_TxHalfCpltCallback\(\)*](#)
- [*HAL_I2S_TxCpltCallback\(\)*](#)
- [*HAL_I2S_RxHalfCpltCallback\(\)*](#)
- [*HAL_I2S_RxCpltCallback\(\)*](#)
- [*HAL_I2S_ErrorCallback\(\)*](#)

23.2.4 Peripheral State and Errors functions

This subsection permits to get in run-time the status of the peripheral and the data flow.

This section contains the following APIs:

- [*HAL_I2S_GetState\(\)*](#)
- [*HAL_I2S_GetError\(\)*](#)

23.2.5 Detailed description of functions

HAL_I2S_Init

Function name	HAL_StatusTypeDef HAL_I2S_Init (I2S_HandleTypeDef * hi2s)
Function description	Initializes the I2S according to the specified parameters in the I2S_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2S_DeInit

Function name	HAL_StatusTypeDef HAL_I2S_DeInit (I2S_HandleTypeDef * hi2s)
Function description	Deinitializes the I2S peripheral.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module

Return values

- **HAL:** status

HAL_I2S_MspInit

Function name	void HAL_I2S_MspInit (I2S_HandleTypeDef * hi2s)
Function description	I2S MSP Init.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • None:

HAL_I2S_MspDelInit

Function name	void HAL_I2S_MspDelInit (I2S_HandleTypeDef * hi2s)
Function description	I2S MSP DelInit.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • None:

HAL_I2S_Transmit

Function name	HAL_StatusTypeDef HAL_I2S_Transmit (I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Transmit an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module • pData: a 16-bit pointer to data buffer. • Size: number of data sample to be sent: • Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When a 16-bit data frame or a 16-bit data frame extended is selected during the I2S configuration phase, the Size parameter means the number of 16-bit data length in the transaction and when a 24-bit data frame or a 32-bit data frame is selected the Size parameter means the number of 16-bit data length. • The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).

HAL_I2S_Receive

Function name	HAL_StatusTypeDef HAL_I2S_Receive (I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Receive an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module • pData: a 16-bit pointer to data buffer. • Size: number of data sample to be sent: • Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When a 16-bit data frame or a 16-bit data frame extended is selected during the I2S configuration phase, the Size parameter means the number of 16-bit data length in the transaction and when a 24-bit data frame or a 32-bit data frame is selected the Size parameter means the number of 16-bit data length. • The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).

- In I2S Master Receiver mode, just after enabling the peripheral the clock will be generated in continuous way and as the I2S is not disabled at the end of the I2S transaction.

HAL_I2S_Transmit_IT

Function name	HAL_StatusTypeDef HAL_I2S_Transmit_IT (I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size)
Function description	Transmit an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module • pData: a 16-bit pointer to data buffer. • Size: number of data sample to be sent:
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When a 16-bit data frame or a 16-bit data frame extended is selected during the I2S configuration phase, the Size parameter means the number of 16-bit data length in the transaction and when a 24-bit data frame or a 32-bit data frame is selected the Size parameter means the number of 16-bit data length. • The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).

HAL_I2S_Receive_IT

Function name	HAL_StatusTypeDef HAL_I2S_Receive_IT (I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size)
Function description	Receive an amount of data in non-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module • pData: a 16-bit pointer to the Receive data buffer. • Size: number of data sample to be sent:
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When a 16-bit data frame or a 16-bit data frame extended is selected during the I2S configuration phase, the Size parameter means the number of 16-bit data length in the transaction and when a 24-bit data frame or a 32-bit data frame is selected the Size parameter means the number of 16-bit data length. • The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming). • It is recommended to use DMA for the I2S receiver to avoid de-synchronization between Master and Slave otherwise the I2S interrupt should be optimized.

HAL_I2S_IRQHandler

Function name	void HAL_I2S_IRQHandler (I2S_HandleTypeDef * hi2s)
Function description	This function handles I2S interrupt request.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • None:

HAL_I2S_Transmit_DMA

Function name	HAL_StatusTypeDef HAL_I2S_Transmit_DMA (I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size)
Function description	Transmit an amount of data in non-blocking mode with DMA.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module • pData: a 16-bit pointer to the Transmit data buffer. • Size: number of data sample to be sent:
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When a 16-bit data frame or a 16-bit data frame extended is selected during the I2S configuration phase, the Size parameter means the number of 16-bit data length in the transaction and when a 24-bit data frame or a 32-bit data frame is selected the Size parameter means the number of 16-bit data length. • The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).

HAL_I2S_Receive_DMA

Function name	HAL_StatusTypeDef HAL_I2S_Receive_DMA (I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size)
Function description	Receive an amount of data in non-blocking mode with DMA.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module • pData: a 16-bit pointer to the Receive data buffer. • Size: number of data sample to be sent:
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When a 16-bit data frame or a 16-bit data frame extended is selected during the I2S configuration phase, the Size parameter means the number of 16-bit data length in the transaction and when a 24-bit data frame or a 32-bit data frame is selected the Size parameter means the number of 16-bit data length. • The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).

HAL_I2S_DMAPause

Function name	HAL_StatusTypeDef HAL_I2S_DM_PAUSE (I2S_HandleTypeDef * hi2s)
Function description	Pauses the audio stream playing from the Media.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2S_DMAResume

Function name	HAL_StatusTypeDef HAL_I2S_DMAResume (I2S_HandleTypeDef * hi2s)
Function description	Resumes the audio stream playing from the Media.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2S_DMAStop

Function name	HAL_StatusTypeDef HAL_I2S_DMAStop (I2S_HandleTypeDef * hi2s)
Function description	Resumes the audio stream playing from the Media.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_I2S_TxHalfCpltCallback

Function name	void HAL_I2S_TxHalfCpltCallback (I2S_HandleTypeDef * hi2s)
Function description	Tx Transfer Half completed callbacks.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • None:

HAL_I2S_TxCpltCallback

Function name	void HAL_I2S_TxCpltCallback (I2S_HandleTypeDef * hi2s)
Function description	Tx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • None:

HAL_I2S_RxHalfCpltCallback

Function name	void HAL_I2S_RxHalfCpltCallback (I2S_HandleTypeDef * hi2s)
---------------	---

Function description	Rx Transfer half completed callbacks.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • None:

HAL_I2S_RxCpltCallback

Function name	void HAL_I2S_RxCpltCallback (I2S_HandleTypeDef * hi2s)
Function description	Rx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • None:

HAL_I2S_ErrorCallback

Function name	void HAL_I2S_ErrorCallback (I2S_HandleTypeDef * hi2s)
Function description	I2S error callbacks.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • None:

HAL_I2S_GetState

Function name	HAL_I2S_StateTypeDef HAL_I2S_GetState (I2S_HandleTypeDef * hi2s)
Function description	Return the I2S state.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_I2S_GetError

Function name	uint32_t HAL_I2S_GetError (I2S_HandleTypeDef * hi2s)
Function description	Return the I2S error code.
Parameters	<ul style="list-style-type: none"> • hi2s: pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module
Return values	<ul style="list-style-type: none"> • I2S: Error Code

23.3 I2S Firmware driver defines

23.3.1 I2S

I2S Audio Frequency

I2S_AUDIOFREQ_192K

I2S_AUDIOFREQ_96K



I2S_AUDIOFREQ_48K
I2S_AUDIOFREQ_44K
I2S_AUDIOFREQ_32K
I2S_AUDIOFREQ_22K
I2S_AUDIOFREQ_16K
I2S_AUDIOFREQ_11K
I2S_AUDIOFREQ_8K
I2S_AUDIOFREQ_DEFAULT
IS_I2S_AUDIO_FREQ

I2S Clock Polarity

I2S_CPOL_LOW
I2S_CPOL_HIGH
IS_I2S_CPOL

I2S Data Format

I2S_DATAFORMAT_16B
I2S_DATAFORMAT_16B_EXTENDED
I2S_DATAFORMAT_24B
I2S_DATAFORMAT_32B
IS_I2S_DATA_FORMAT

I2S Error Codes

HAL_I2S_ERROR_NONE	No error
HAL_I2S_ERROR_UDR	I2S Underrun error
HAL_I2S_ERROR_OVR	I2S Overrun error
HAL_I2S_ERROR_FRE	I2S Frame format error
HAL_I2S_ERROR_DMA	DMA transfer error

I2S Exported Macros

<code>_HAL_I2S_RESET_HANDLE_STATE</code>	Description: <ul style="list-style-type: none">Reset I2S handle state. Parameters: <ul style="list-style-type: none"><code>_HANDLE_</code>: specifies the I2S Handle. Return value: <ul style="list-style-type: none">None
<code>_HAL_I2S_ENABLE</code>	Description: <ul style="list-style-type: none">Enable or disable the specified SPI peripheral (in I2S mode). Parameters:

- `__HANDLE__`: specifies the I2S Handle.

Return value:

- None

`__HAL_I2S_DISABLE`

`__HAL_I2S_ENABLE_IT`

Description:

- Enable or disable the specified I2S interrupts.

Parameters:

- `__HANDLE__`: specifies the I2S Handle.
- `__INTERRUPT__`: specifies the interrupt source to enable or disable. This parameter can be one of the following values:
 - `I2S_IT_TXE`: Tx buffer empty interrupt enable
 - `I2S_IT_RXNE`: RX buffer not empty interrupt enable
 - `I2S_IT_ERR`: Error interrupt enable

Return value:

- None

`__HAL_I2S_DISABLE_IT`

`__HAL_I2S_GET_IT_SOURCE`

Description:

- Checks if the specified I2S interrupt source is enabled or disabled.

Parameters:

- `__HANDLE__`: specifies the I2S Handle. This parameter can be `I2S` where `x`: 1, 2, or 3 to select the I2S peripheral.
- `__INTERRUPT__`: specifies the I2S interrupt source to check. This parameter can be one of the following values:
 - `I2S_IT_TXE`: Tx buffer empty interrupt enable
 - `I2S_IT_RXNE`: RX buffer not empty interrupt enable
 - `I2S_IT_ERR`: Error interrupt enable

Return value:

- The: new state of `__IT__` (TRUE or FALSE).

Description:

- Checks whether the specified I2S flag is set or not.

Parameters:

- `__HANDLE__`: specifies the I2S Handle.
- `__FLAG__`: specifies the flag to check. This parameter can be one of the following

values:

- I2S_FLAG_RXNE: Receive buffer not empty flag
- I2S_FLAG_TXE: Transmit buffer empty flag
- I2S_FLAG_UDR: Underrun flag
- I2S_FLAG_OVR: Overrun flag
- I2S_FLAG_FRE: Frame error flag
- I2S_FLAG_CHSIDE: Channel Side flag
- I2S_FLAG_BSY: Busy flag

Return value:

- The new state of __FLAG__ (TRUE or FALSE).

`_HAL_I2S_CLEAR_OVRFAG`

Description:

- Clears the I2S OVR pending flag.

Parameters:

- `_HANDLE_`: specifies the I2S Handle.

Return value:

- None

`_HAL_I2S_CLEAR_UDRFLAG`

Description:

- Clears the I2S UDR pending flag.

Parameters:

- `_HANDLE_`: specifies the I2S Handle.

Return value:

- None

I2S Flag definition

`I2S_FLAG_TXE`

`I2S_FLAG_RXNE`

`I2S_FLAG_UDR`

`I2S_FLAG_OVR`

`I2S_FLAG_FRE`

`I2S_FLAG_CHSIDE`

`I2S_FLAG_BSY`

I2S Interrupt configuration definition

`I2S_IT_TXE`

`I2S_IT_RXNE`

`I2S_IT_ERR`

I2S MCLK Output

`I2S_MCLKOUTPUT_ENABLE`

I2S_MCLKOUTPUT_DISABLE

IS_I2S_MCLK_OUTPUT

I2S Mode

I2S_MODE_SLAVE_TX

I2S_MODE_SLAVE_RX

I2S_MODE_MASTER_TX

I2S_MODE_MASTER_RX

IS_I2S_MODE

I2S Standard

I2S_STANDARD_PHILIPS

I2S_STANDARD_MSB

I2S_STANDARD_LSB

I2S_STANDARD_PCM_SHORT

I2S_STANDARD_PCM_LONG

IS_I2S_STANDARD

24 HAL IRDA Generic Driver

24.1 IRDA Firmware driver registers structures

24.1.1 IRDA_InitTypeDef

Data Fields

- *uint32_t BaudRate*
- *uint32_t WordLength*
- *uint32_t Parity*
- *uint32_t Mode*
- *uint8_t Prescaler*
- *uint32_t IrDAMode*

Field Documentation

- ***uint32_t IRDA_InitTypeDef::BaudRate***

This member configures the IRDA communication baud rate. The baud rate is computed using the following formula:
 $\text{IntegerDivider} = ((\text{PCLKx}) / (16 * (\text{hirda->Init.BaudRate})))$
 $\text{FractionalDivider} = ((\text{IntegerDivider} - ((\text{uint32_t}) \text{ IntegerDivider})) * 16) + 0.5$

- ***uint32_t IRDA_InitTypeDef::WordLength***

Specifies the number of data bits transmitted or received in a frame. This parameter can be a value of [IRDA_Word_Length](#)

- ***uint32_t IRDA_InitTypeDef::Parity***

Specifies the parity mode. This parameter can be a value of [IRDA_Parity](#)
Note:When parity is enabled, the computed parity is inserted at the MSB position of the transmitted data (9th bit when the word length is set to 9 data bits; 8th bit when the word length is set to 8 data bits).

- ***uint32_t IRDA_InitTypeDef::Mode***

Specifies whether the Receive or Transmit mode is enabled or disabled. This parameter can be a value of [IRDA_Transfer_Mode](#)

- ***uint8_t IRDA_InitTypeDef::Prescaler***

Specifies the Prescaler value prescaler value to be programmed in the IrDA low-power Baud Register, for defining pulse width on which burst acceptance/rejection will be decided. This value is used as divisor of system clock to achieve required pulse width.

- ***uint32_t IRDA_InitTypeDef::IrDAMode***

Specifies the IrDA mode This parameter can be a value of [IRDA_Low_Power](#)

24.1.2 IRDA_HandleTypeDef

Data Fields

- *USART_TypeDef * Instance*
- *IRDA_InitTypeDef Init*
- *uint8_t * pTxBuffPtr*
- *uint16_t TxXferSize*
- *uint16_t TxXferCount*
- *uint8_t * pRxBuffPtr*
- *uint16_t RxXferSize*
- *uint16_t RxXferCount*
- *DMA_HandleTypeDef * hdmatx*
- *DMA_HandleTypeDef * hdmarx*

- ***HAL_LockTypeDef Lock***
- ***_IO HAL_IRDA_StateTypeDef State***
- ***_IO uint32_t ErrorCode***

Field Documentation

- ***USART_TypeDef* IRDA_HandleTypeDef::Instance***
USART registers base address
- ***IRDA_InitTypeDef IRDA_HandleTypeDef::Init***
IRDA communication parameters
- ***uint8_t* IRDA_HandleTypeDef::pTxBuffPtr***
Pointer to IRDA Tx transfer Buffer
- ***uint16_t IRDA_HandleTypeDef::TxXferSize***
IRDA Tx Transfer size
- ***uint16_t IRDA_HandleTypeDef::TxXferCount***
IRDA Tx Transfer Counter
- ***uint8_t* IRDA_HandleTypeDef::pRxBuffPtr***
Pointer to IRDA Rx transfer Buffer
- ***uint16_t IRDA_HandleTypeDef::RxXferSize***
IRDA Rx Transfer size
- ***uint16_t IRDA_HandleTypeDef::RxXferCount***
IRDA Rx Transfer Counter
- ***DMA_HandleTypeDef* IRDA_HandleTypeDef::hdmatx***
IRDA Tx DMA Handle parameters
- ***DMA_HandleTypeDef* IRDA_HandleTypeDef::hdmarx***
IRDA Rx DMA Handle parameters
- ***HAL_LockTypeDef IRDA_HandleTypeDef::Lock***
Locking object
- ***_IO HAL_IRDA_StateTypeDef IRDA_HandleTypeDef::State***
IRDA communication state
- ***_IO uint32_t IRDA_HandleTypeDef::ErrorCode***
IRDA Error code

24.2 IRDA Firmware driver API description

24.2.1 How to use this driver

The IRDA HAL driver can be used as follows:

1. Declare a IRDA_HandleTypeDef handle structure.
2. Initialize the IRDA low level resources by implementing the HAL_IRDA_MspInit() API:
 - a. Enable the USARTx interface clock.
 - b. IRDA pins configuration:
 - Enable the clock for the IRDA GPIOs.
 - Configure the IRDA pins as alternate function pull-up.
 - c. NVIC configuration if you need to use interrupt process
(HAL_IRDA_Transmit_IT() and HAL_IRDA_Receive_IT() APIs):
 - Configure the USARTx interrupt priority.
 - Enable the NVIC USART IRQ handle.
 - d. DMA Configuration if you need to use DMA process
(HAL_IRDA_Transmit_DMA() and HAL_IRDA_Receive_DMA() APIs):
 - Declare a DMA handle structure for the Tx/Rx channel.
 - Enable the DMAx interface clock.
 - Configure the declared DMA handle structure with the required Tx/Rx parameters.
 - Configure the DMA Tx/Rx channel.

- Associate the initialized DMA handle to the IRDA DMA Tx/Rx handle.
 - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx/Rx channel.
 - Configure the USARTx interrupt priority and enable the NVIC USART IRQ handle (used for last byte sending completion detection in DMA non circular mode)
3. Program the Baud Rate, Word Length, Parity, IrDA Mode, Prescaler and Mode(Receiver/Transmitter) in the hirda Init structure.
 4. Initialize the IRDA registers by calling the HAL_IRDA_Init() API:
 - This API configures also the low level Hardware GPIO, CLOCK, CORTEX...etc by calling the customized HAL_IRDA_MspInit() API. The specific IRDA interrupts (Transmission complete interrupt, RXNE interrupt and Error Interrupts) will be managed using the macros __HAL_IRDA_ENABLE_IT() and __HAL_IRDA_DISABLE_IT() inside the transmit and receive process.
 5. Three operation modes are available within this driver :

Polling mode IO operation

- Send an amount of data in blocking mode using HAL_IRDA_Transmit()
- Receive an amount of data in blocking mode using HAL_IRDA_Receive()

Interrupt mode IO operation

- Send an amount of data in non blocking mode using HAL_IRDA_Transmit_IT()
- At transmission end of transfer HAL_IRDA_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_IRDA_TxCpltCallback
- Receive an amount of data in non blocking mode using HAL_IRDA_Receive_IT()
- At reception end of transfer HAL_IRDA_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_IRDA_RxCpltCallback
- In case of transfer Error, HAL_IRDA_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_IRDA_ErrorCallback

DMA mode IO operation

- Send an amount of data in non blocking mode (DMA) using HAL_IRDA_Transmit_DMA()
- At transmission end of transfer HAL_IRDA_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_IRDA_TxCpltCallback
- Receive an amount of data in non blocking mode (DMA) using HAL_IRDA_Receive_DMA()
- At reception end of transfer HAL_IRDA_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_IRDA_RxCpltCallback
- In case of transfer Error, HAL_IRDA_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_IRDA_ErrorCallback

IRDA HAL driver macros list

Below the list of most used macros in IRDA HAL driver.

- __HAL_IRDA_ENABLE: Enable the IRDA peripheral
- __HAL_IRDA_DISABLE: Disable the IRDA peripheral
- __HAL_IRDA_GET_FLAG : Check whether the specified IRDA flag is set or not
- __HAL_IRDA_CLEAR_FLAG : Clear the specified IRDA pending flag
- __HAL_IRDA_ENABLE_IT: Enable the specified IRDA interrupt
- __HAL_IRDA_DISABLE_IT: Disable the specified IRDA interrupt

- `_HAL_IRDA_GET_IT_SOURCE`: Check whether the specified IRDA interrupt has occurred or not



You can refer to the IRDA HAL driver header file for more useful macros

24.2.2 Initialization and Configuration functions

This subsection provides a set of functions allowing to initialize the USARTx or the UARTy in IrDA mode.

- For the asynchronous mode only these parameters can be configured:
 - Baud Rate
 - Word Length
 - Parity
 - Prescaler: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected. The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).
 - Mode: Receiver/transmitter modes
 - IrDAMode: the IrDA can operate in the Normal mode or in the Low power mode.

The `HAL_IRDA_Init()` function follows IRDA configuration procedures (details for the procedures are available in reference manual (RM0038)).

This section contains the following APIs:

- `HAL_IRDA_Init\(\)`
- `HAL_IRDA_DeInit\(\)`
- `HAL_IRDA_MspInit\(\)`
- `HAL_IRDA_MspDeInit\(\)`

24.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the IRDA data transfers.

IrDA is a half duplex communication protocol. If the Transmitter is busy, any data on the IrDA receive line will be ignored by the IrDA decoder and if the Receiver is busy, data on the TX from the USART to IrDA will not be encoded by IrDA. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

1. There are two modes of transfer:
 - Blocking mode: The communication is performed in polling mode. The HAL status of all data processing is returned by the same function after finishing transfer.
 - No-Blocking mode: The communication is performed using Interrupts or DMA, These API's return the HAL status. The end of the data processing will be indicated through the dedicated IRDA IRQ when using Interrupt mode or the DMA IRQ when using DMA mode. The `HAL_IRDA_TxCpltCallback()`, `HAL_IRDA_RxCpltCallback()` user callbacks will be executed respectively at the end of the transmit or Receive process The `HAL_IRDA_ErrorCallback()` user callback will be executed when a communication error is detected
2. Blocking mode APIs are :
 - `HAL_IRDA_Transmit()`
 - `HAL_IRDA_Receive()`
3. Non Blocking mode APIs with Interrupt are :

- HAL_IRDA_Transmit_IT()
 - HAL_IRDA_Receive_IT()
 - HAL_IRDA_IRQHandler()
4. Non Blocking mode functions with DMA are :
- HAL_IRDA_Transmit_DMA()
 - HAL_IRDA_Receive_DMA()
 - HAL_IRDA_DMAPause()
 - HAL_IRDA_DMAResume()
 - HAL_IRDA_DMAStop()
5. A set of Transfer Complete Callbacks are provided in non Blocking mode:
- HAL_IRDA_TxHalfCpltCallback()
 - HAL_IRDA_TxCpltCallback()
 - HAL_IRDA_RxHalfCpltCallback()
 - HAL_IRDA_RxCpltCallback()
 - HAL_IRDA_ErrorCallback()

This section contains the following APIs:

- [*HAL_IRDA_Transmit\(\)*](#)
- [*HAL_IRDA_Receive\(\)*](#)
- [*HAL_IRDA_Transmit_IT\(\)*](#)
- [*HAL_IRDA_Receive_IT\(\)*](#)
- [*HAL_IRDA_Transmit_DMA\(\)*](#)
- [*HAL_IRDA_Receive_DMA\(\)*](#)
- [*HAL_IRDA_DM_PAUSE\(\)*](#)
- [*HAL_IRDA_DMAResume\(\)*](#)
- [*HAL_IRDA_DMAStop\(\)*](#)
- [*HAL_IRDA_IRQHandler\(\)*](#)
- [*HAL_IRDA_TxCpltCallback\(\)*](#)
- [*HAL_IRDA_TxHalfCpltCallback\(\)*](#)
- [*HAL_IRDA_RxCpltCallback\(\)*](#)
- [*HAL_IRDA_RxHalfCpltCallback\(\)*](#)
- [*HAL_IRDA_ErrorCallback\(\)*](#)

24.2.4 Peripheral State and Errors functions

This subsection provides a set of functions allowing to return the State of IrDA communication process and also return Peripheral Errors occurred during communication process

- HAL_IRDA_GetState() API can be helpful to check in run-time the state of the IRDA peripheral.
- HAL_IRDA_GetError() check in run-time errors that could be occurred during communication.

This section contains the following APIs:

- [*HAL_IRDA_GetState\(\)*](#)
- [*HAL_IRDA_GetError\(\)*](#)

24.2.5 Detailed description of functions

HAL_IRDA_Init

Function name	HAL_StatusTypeDef HAL_IRDA_Init (IRDA_HandleTypeDef * hirda)
---------------	---

Function description	Initializes the IRDA mode according to the specified parameters in the IRDA_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> HAL: status

HAL_IRDA_DeInit

Function name	HAL_StatusTypeDef HAL_IRDA_DeInit (IRDA_HandleTypeDef * hirda)
Function description	DeInitializes the IRDA peripheral.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> HAL: status

HAL_IRDA_MspInit

Function name	void HAL_IRDA_MspInit (IRDA_HandleTypeDef * hirda)
Function description	IRDA MSP Init.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> None:

HAL_IRDA_MspDelInit

Function name	void HAL_IRDA_MspDelInit (IRDA_HandleTypeDef * hirda)
Function description	IRDA MSP DelInit.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> None:

HAL_IRDA_Transmit

Function name	HAL_StatusTypeDef HAL_IRDA_Transmit (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Sends an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module. pData: Pointer to data buffer Size: Amount of data to be sent Timeout: Specify timeout value

Return values	<ul style="list-style-type: none"> HAL: status
---------------	--

HAL_IRDA_Receive

Function name	HAL_StatusTypeDef HAL_IRDA_Receive (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Receive an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module. pData: Pointer to data buffer Size: Amount of data to be received Timeout: Specify timeout value
Return values	<ul style="list-style-type: none"> HAL: status

HAL_IRDA_Transmit_IT

Function name	HAL_StatusTypeDef HAL_IRDA_Transmit_IT (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size)
Function description	Sends an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module. pData: Pointer to data buffer Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> HAL: status

HAL_IRDA_Receive_IT

Function name	HAL_StatusTypeDef HAL_IRDA_Receive_IT (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size)
Function description	Receives an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module. pData: Pointer to data buffer Size: Amount of data to be received
Return values	<ul style="list-style-type: none"> HAL: status

HAL_IRDA_Transmit_DMA

Function name	HAL_StatusTypeDef HAL_IRDA_Transmit_DMA (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size)
Function description	Sends an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.

- **pData:** Pointer to data buffer
 - **Size:** Amount of data to be sent
- Return values
- **HAL:** status

HAL_IRDA_Receive_DMA

Function name	HAL_StatusTypeDef HAL_IRDA_Receive_DMA (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size)
Function description	Receive an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> • hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module. • pData: Pointer to data buffer • Size: Amount of data to be received
Return values	• HAL: status
Notes	• When the IRDA parity is enabled (PCE = 1) the data received contain the parity bit.

HAL_IRDA_DMAPause

Function name	HAL_StatusTypeDef HAL_IRDA_DMAPause (IRDA_HandleTypeDef * hirda)
Function description	Pauses the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	• HAL: status

HAL_IRDA_DMAResume

Function name	HAL_StatusTypeDef HAL_IRDA_DMAResume (IRDA_HandleTypeDef * hirda)
Function description	Resumes the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	• HAL: status

HAL_IRDA_DMAStop

Function name	HAL_StatusTypeDef HAL_IRDA_DMAStop (IRDA_HandleTypeDef * hirda)
Function description	Stops the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified UART module.

Return values	<ul style="list-style-type: none"> HAL: status
---------------	--

HAL_IRDA_IRQHandler

Function name	void HAL_IRDA_IRQHandler (IRDA_HandleTypeDef * hirda)
Function description	This function handles IRDA interrupt request.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> None:

HAL_IRDA_TxCpltCallback

Function name	void HAL_IRDA_TxCpltCallback (IRDA_HandleTypeDef * hirda)
Function description	Tx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> None:

HAL_IRDA_RxCpltCallback

Function name	void HAL_IRDA_RxCpltCallback (IRDA_HandleTypeDef * hirda)
Function description	Rx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> None:

HAL_IRDA_TxHalfCpltCallback

Function name	void HAL_IRDA_TxHalfCpltCallback (IRDA_HandleTypeDef * hirda)
Function description	Tx Half Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	<ul style="list-style-type: none"> None:

HAL_IRDA_RxHalfCpltCallback

Function name	void HAL_IRDA_RxHalfCpltCallback (IRDA_HandleTypeDef * hirda)
Function description	Rx Half Transfer complete callbacks.

Parameters	<ul style="list-style-type: none"> • hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> • None:

HAL_IRDA_ErrorCallback

Function name	void HAL_IRDA_ErrorCallback (IRDA_HandleTypeDef * hirda)
Function description	IRDA error callbacks.
Parameters	<ul style="list-style-type: none"> • hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> • None:

HAL_IRDA_GetState

Function name	HAL_IRDA_StateTypeDef HAL_IRDA_GetState (IRDA_HandleTypeDef * hirda)
Function description	Returns the IRDA state.
Parameters	<ul style="list-style-type: none"> • hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_IRDA_GetError

Function name	uint32_t HAL_IRDA_GetError (IRDA_HandleTypeDef * hirda)
Function description	Return the IRDA error code.
Parameters	<ul style="list-style-type: none"> • hirda: Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.
Return values	<ul style="list-style-type: none"> • IRDA: Error Code

24.3 IRDA Firmware driver defines

24.3.1 IRDA

IRDA Error Codes

HAL_IRDA_ERROR_NONE	No error
HAL_IRDA_ERROR_PE	Parity error
HAL_IRDA_ERROR_NE	Noise error
HAL_IRDA_ERROR_FE	frame error
HAL_IRDA_ERROR_ORE	Overrun error
HAL_IRDA_ERROR_DMA	DMA transfer error

IRDA Exported Macros



[__HAL_IRDA_RESET_HANDLE_STATE](#)**Description:**

- Reset IRDA handle state.

Parameters:

- [__HANDLE__](#): specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

[__HAL_IRDA_FLUSH_DRREGISTER](#)**Description:**

- Flush the IRDA DR register.

Parameters:

- [__HANDLE__](#): specifies the USART Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

[__HAL_IRDA_GET_FLAG](#)**Description:**

- Check whether the specified IRDA flag is set or not.

Parameters:

- [__HANDLE__](#): specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).
- [__FLAG__](#): specifies the flag to check. This parameter can be one of the following values:
 - [IRDA_FLAG_TXE](#): Transmit data register empty flag
 - [IRDA_FLAG_TC](#): Transmission Complete flag
 - [IRDA_FLAG_RXNE](#): Receive data register not empty flag
 - [IRDA_FLAG_IDLE](#): Idle Line detection flag
 - [IRDA_FLAG_ORE](#): OverRun Error flag
 - [IRDA_FLAG_NE](#): Noise Error flag
 - [IRDA_FLAG_FE](#): Framing Error flag
 - [IRDA_FLAG_PE](#): Parity Error flag

Return value:

- The: new state of __FLAG__ (TRUE or FALSE).

__HAL_IRDA_CLEAR_FLAG

Description:

- Clear the specified IRDA pending flag.

Parameters:

- __HANDLE__: specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).
- __FLAG__: specifies the flag to check. This parameter can be any combination of the following values:
 - IRDA_FLAG_TC: Transmission Complete flag.
 - IRDA_FLAG_RXNE: Receive data register not empty flag.

Return value:

- None

Notes:

- PE (Parity error), FE (Framing error), NE (Noise error), ORE (OverRun error) and IDLE (Idle line detected) flags are cleared by software sequence: a read operation to USART_SR register followed by a read operation to USART_DR register. RXNE flag can be also cleared by a read to the USART_DR register. TC flag can be also cleared by software sequence: a read operation to USART_SR register followed by a write operation to USART_DR register. TXE flag is cleared only by a write to the USART_DR register.

__HAL_IRDA_CLEAR_PEFLAG

Description:

- Clear the IRDA PE pending flag.

Parameters:

- __HANDLE__: specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

[__HAL_IRDA_CLEAR_FEFLAG](#)**Description:**

- Clear the IRDA FE pending flag.

Parameters:

- [__HANDLE__](#): specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

[__HAL_IRDA_CLEAR_NEFLAG](#)**Description:**

- Clear the IRDA NE pending flag.

Parameters:

- [__HANDLE__](#): specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

[__HAL_IRDA_CLEAR_OREFLAG](#)**Description:**

- Clear the IRDA ORE pending flag.

Parameters:

- [__HANDLE__](#): specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

[__HAL_IRDA_CLEAR_IDLEFLAG](#)**Description:**

- Clear the IRDA IDLE pending flag.

Parameters:

- [__HANDLE__](#): specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

[__HAL_IRDA_ENABLE_IT](#)**Description:**

- Enable the specified IRDA interrupt.

Parameters:

- __HANDLE__: specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).
- __INTERRUPT__: specifies the IRDA interrupt source to enable. This parameter can be one of the following values:
 - IRDA_IT_TXE: Transmit Data Register empty interrupt
 - IRDA_IT_TC: Transmission complete interrupt
 - IRDA_IT_RXNE: Receive Data register not empty interrupt
 - IRDA_IT_IDLE: Idle line detection interrupt
 - IRDA_IT_PE: Parity Error interrupt
 - IRDA_IT_ERR: Error interrupt(Frame error, noise error, overrun error)

Return value:

- None

__HAL_IRDA_DISABLE_IT

- Disable the specified IRDA interrupt.

Parameters:

- __HANDLE__: specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).
- __INTERRUPT__: specifies the IRDA interrupt source to disable. This parameter can be one of the following values:
 - IRDA_IT_TXE: Transmit Data Register empty interrupt
 - IRDA_IT_TC: Transmission complete interrupt
 - IRDA_IT_RXNE: Receive Data register not empty interrupt
 - IRDA_IT_IDLE: Idle line detection interrupt
 - IRDA_IT_PE: Parity Error interrupt
 - IRDA_IT_ERR: Error interrupt(Frame error, noise

error, overrun error)

Return value:

- None

`_HAL_IRDA_GET_IT_SOURCE`

Description:

- Check whether the specified IRDA interrupt has occurred or not.

Parameters:

- `_HANDLE_`: specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).
- `_IT_`: specifies the IRDA interrupt source to check. This parameter can be one of the following values:
 - `IRDA_IT_TXE`: Transmit Data Register empty interrupt
 - `IRDA_IT_TC`: Transmission complete interrupt
 - `IRDA_IT_RXNE`: Receive Data register not empty interrupt
 - `IRDA_IT_IDLE`: Idle line detection interrupt
 - `IRDA_IT_ERR`: Error interrupt
 - `IRDA_IT_PE`: Parity Error interrupt

Return value:

- The: new state of `_IT_` (TRUE or FALSE).

`_HAL_IRDA_ONE_BIT_SAMPLE_ENABLE`

Description:

- Enables the IRDA one bit sample method.

Parameters:

- `_HANDLE_`: specifies the IRDA Handle.

Return value:

- None

`_HAL_IRDA_ONE_BIT_SAMPLE_DISABLE`

Description:

- Disables the IRDA one bit sample method.

Parameters:

- `_HANDLE_`: specifies the IRDA Handle.

Return value:

- None

`_HAL_IRDA_ENABLE`

Description:

- Enable UART/USART associated to IRDA Handle.

Parameters:

- `_HANDLE_`: specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

`_HAL_IRDA_DISABLE`

Description:

- Disable UART/USART associated to IRDA Handle.

Parameters:

- `_HANDLE_`: specifies the IRDA Handle. IRDA Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

IRDA Flags

`IRDA_FLAG_TXE`

`IRDA_FLAG_TC`

`IRDA_FLAG_RXNE`

`IRDA_FLAG_IDLE`

`IRDA_FLAG_ORE`

`IRDA_FLAG_NE`

`IRDA_FLAG_FE`

`IRDA_FLAG_PE`

IRDA Interrupt Definitions

`IRDA_IT_PE`

`IRDA_IT_TXE`

`IRDA_IT_TC`

`IRDA_IT_RXNE`

`IRDA_IT_IDLE`

`IRDA_IT_LBD`

IRDA_IT_CTS

IRDA_IT_ERR

IRDA Low Power

IRDA_POWERMODE_LOWPOWER

IRDA_POWERMODE_NORMAL

IRDA One Bit Sampling

IRDA_ONE_BIT_SAMPLE_DISABLE

IRDA_ONE_BIT_SAMPLE_ENABLE

IRDA Parity

IRDA_PARITY_NONE

IRDA_PARITY_EVEN

IRDA_PARITY_ODD

IRDA Transfer Mode

IRDA_MODE_RX

IRDA_MODE_TX

IRDA_MODE_TX_RX

IRDA Word Length

IRDA_WORDLENGTH_8B

IRDA_WORDLENGTH_9B

25 HAL IWDG Generic Driver

25.1 IWDG Firmware driver registers structures

25.1.1 IWDG_InitTypeDef

Data Fields

- *uint32_t Prescaler*
- *uint32_t Reload*

Field Documentation

- *uint32_t IWDG_InitTypeDef::Prescaler*
Select the prescaler of the IWDG. This parameter can be a value of [IWDG_Prescaler](#)
- *uint32_t IWDG_InitTypeDef::Reload*
Specifies the IWDG down-counter reload value. This parameter must be a number between Min_Data = 0 and Max_Data = 0x0FFF

25.1.2 IWDG_HandleTypeDef

Data Fields

- *IWDG_TypeDef * Instance*
- *IWDG_InitTypeDef Init*

Field Documentation

- *IWDG_TypeDef* IWDG_HandleTypeDef::Instance*
Register base address
- *IWDG_InitTypeDef IWDG_HandleTypeDef::Init*
IWDG required parameters

25.2 IWDG Firmware driver API description

25.2.1 IWDG Generic features

- The IWDG can be started by either software or hardware (configurable through option byte).
- The IWDG is clocked by Low-Speed clock (LSI) and thus stays active even if the main clock fails.
- Once the IWDG is started, the LSI is forced ON and both can not be disabled. The counter starts counting down from the reset value (0xFFFF). When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).
- Whenever the key value 0x0000 AAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.
- The IWDG is implemented in the VDD voltage domain that is still functional in STOP and STANDBY mode (IWDG reset can wake-up from STANDBY). IWDGRST flag in RCC_CSR register can be used to inform when an IWDG reset occurs.
- Debug mode : When the microcontroller enters debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module, accessible through
`_HAL_DBGMCU_FREEZE_IWDG()` and `_HAL_DBGMCU_UNFREEZE_IWDG()` macros

- Min-max timeout value @37KHz (LSI): ~108us / ~28.3s The IWDG timeout may vary due to LSI frequency dispersion. STM32L1xx devices provide the capability to measure the LSI frequency (LSI clock connected internally to TIM10 CH1 input capture). The measured value can be used to have an IWDG timeout with an acceptable accuracy. For more information, please refer to the STM32L1xx Reference manual.

25.2.2 How to use this driver

1. Use IWDG using HAL_IWDG_Init() function to :
 - Enable instance by writing Start keyword in IWDG_KEY register. LSI clock is forced ON and IWDG counter starts downcounting.
 - Enable write access to configuration register: IWDG_PR, IWDG_RLR.
 - Configure the IWDG prescaler and counter reload value. This reload value will be loaded in the IWDG counter each time the watchdog is reloaded, then the IWDG will start counting down from this value.
 - wait for status flags to be reset"
2. Then the application program must refresh the IWDG counter at regular intervals during normal operation to prevent an MCU reset, using HAL_IWDG_Refresh() function.

IWDG HAL driver macros list

Below the list of most used macros in IWDG HAL driver:

- __HAL_IWDG_START: Enable the IWDG peripheral
- __HAL_IWDG_RELOAD_COUNTER: Reloads IWDG counter with value defined in the reload register

25.2.3 Initialization and Start functions

This section provides functions allowing to:

- Initialize the IWDG according to the specified parameters in the IWDG_InitTypeDef of associated handle.
- Once initialization is performed in HAL_IWDG_Init function, Watchdog is reloaded in order to exit function with correct time base.

This section contains the following APIs:

- [**HAL_IWDG_Init\(\)**](#)

25.2.4 IO operation functions

This section provides functions allowing to:

- Refresh the IWDG.

This section contains the following APIs:

- [**HAL_IWDG_Refresh\(\)**](#)

25.2.5 Detailed description of functions

HAL_IWDG_Init

Function name **HAL_StatusTypeDef HAL_IWDG_Init (IWDG_HandleTypeDef *
hiwdg)**

Function description Initialize the IWDG according to the specified parameters in the

IWDG_InitTypeDef and start watchdog.

- | | |
|---------------|--|
| Parameters | <ul style="list-style-type: none"> • hiwdg: pointer to a IWDG_HandleTypeDef structure that contains the configuration information for the specified IWDG module. |
| Return values | <ul style="list-style-type: none"> • HAL: status |

HAL_IWDG_Refresh

Function name **HAL_StatusTypeDef HAL_IWDG_Refresh
(IWDG_HandleTypeDef * hiwdg)**

Function description Refresh the IWDG.

- | | |
|---------------|--|
| Parameters | <ul style="list-style-type: none"> • hiwdg: pointer to a IWDG_HandleTypeDef structure that contains the configuration information for the specified IWDG module. |
| Return values | <ul style="list-style-type: none"> • HAL: status |

25.3 IWDG Firmware driver defines

25.3.1 IWDG

IWDG Exported Macros

_HAL_IWDG_START

Description:

- Enable the IWDG peripheral.

Parameters:

- **_HANDLE_**: IWDG handle

Return value:

- None

_HAL_IWDG_RELOAD_COUNTER

Description:

- Reload IWDG counter with value defined in the reload register (write access to IWDG_PR, IWDG_RLR & IWDG_WINR registers disabled).

Parameters:

- **_HANDLE_**: IWDG handle

Return value:

- None

IWDG Prescaler

IWDG_PRESCALER_4 IWDG prescaler set to 4

IWDG_PRESCALER_8 IWDG prescaler set to 8

IWDG_PRESCALER_16 IWDG prescaler set to 16

IWDG_PRESCALER_32 IWDG prescaler set to 32

IWDG_PRESCALER_64 IWDG prescaler set to 64



IWDG_PRESCALER_128 IWDG prescaler set to 128

IWDG_PRESCALER_256 IWDG prescaler set to 256

26 HAL LCD Generic Driver

26.1 LCD Firmware driver registers structures

26.1.1 LCD_InitTypeDef

Data Fields

- *uint32_t Prescaler*
- *uint32_t Divider*
- *uint32_t Duty*
- *uint32_t Bias*
- *uint32_t VoltageSource*
- *uint32_t Contrast*
- *uint32_t DeadTime*
- *uint32_t PulseOnDuration*
- *uint32_t HighDrive*
- *uint32_t BlinkMode*
- *uint32_t BlinkFrequency*
- *uint32_t MuxSegment*

Field Documentation

- *uint32_t LCD_InitTypeDef::Prescaler*
Configures the LCD Prescaler. This parameter can be one value of [*LCD_Prescaler*](#)
- *uint32_t LCD_InitTypeDef::Divider*
Configures the LCD Divider. This parameter can be one value of [*LCD_Divider*](#)
- *uint32_t LCD_InitTypeDef::Duty*
Configures the LCD Duty. This parameter can be one value of [*LCD_Duty*](#)
- *uint32_t LCD_InitTypeDef::Bias*
Configures the LCD Bias. This parameter can be one value of [*LCD_Bias*](#)
- *uint32_t LCD_InitTypeDef::VoltageSource*
Selects the LCD Voltage source. This parameter can be one value of [*LCD_Voltage_Source*](#)
- *uint32_t LCD_InitTypeDef::Contrast*
Configures the LCD Contrast. This parameter can be one value of [*LCD_Contrast*](#)
- *uint32_t LCD_InitTypeDef::DeadTime*
Configures the LCD Dead Time. This parameter can be one value of [*LCD_DeadTime*](#)
- *uint32_t LCD_InitTypeDef::PulseOnDuration*
Configures the LCD Pulse On Duration. This parameter can be one value of [*LCD_PulseOnDuration*](#)
- *uint32_t LCD_InitTypeDef::HighDrive*
Configures the LCD High Drive. This parameter can be one value of [*LCD_HighDrive*](#)
- *uint32_t LCD_InitTypeDef::BlinkMode*
Configures the LCD Blink Mode. This parameter can be one value of [*LCD_BlinkMode*](#)
- *uint32_t LCD_InitTypeDef::BlinkFrequency*
Configures the LCD Blink frequency. This parameter can be one value of [*LCD_BlinkFrequency*](#)
- *uint32_t LCD_InitTypeDef::MuxSegment*
Enable or disable mux segment. This parameter can be set to ENABLE or DISABLE.

26.1.2 LCD_HandleTypeDef

Data Fields

- *LCD_TypeDef * Instance*
- *LCD_InitTypeDef Init*
- *HAL_LockTypeDef Lock*
- *__IO HAL_LCD_StateTypeDef State*
- *__IO uint32_t ErrorCode*

Field Documentation

- *LCD_TypeDef* LCD_HandleTypeDef::Instance*
- *LCD_InitTypeDef LCD_HandleTypeDef::Init*
- *HAL_LockTypeDef LCD_HandleTypeDef::Lock*
- *__IO HAL_LCD_StateTypeDef LCD_HandleTypeDef::State*
- *__IO uint32_t LCD_HandleTypeDef::ErrorCode*

26.2 LCD Firmware driver API description

26.2.1 How to use this driver

The LCD HAL driver can be used as follows:

1. Declare a LCD_HandleTypeDef handle structure.
2. Initialize the LCD low level resources by implement the HAL_LCD_MspInit() API:
 - a. Enable the LCDCLK (same as RTCCLK): to configure the RTCCLK/LCDCLK, proceed as follows:
 - Use RCC function HAL_RCCEx_PeriphCLKConfig in indicating RCC_PERIPHCLK_LCD and selected clock source (HSE, LSI or LSE)
 - The frequency generator allows you to achieve various LCD frame rates starting from an LCD input clock frequency (LCDCLK) which can vary from 32 kHz up to 1 MHz.
 - b. LCD pins configuration:
 - Enable the clock for the LCD GPIOs.
 - Configure these LCD pins as alternate function no-pull.
 - c. Enable the LCD interface clock.
3. Program the Prescaler, Divider, Blink mode, Blink Frequency Duty, Bias, Voltage Source, Dead Time, Pulse On Duration and Contrast in the hlcd Init structure.
4. Initialize the LCD registers by calling the HAL_LCD_Init() API. The HAL_LCD_Init() API configures also the low level Hardware GPIO, CLOCK, ...etc by calling the custumed HAL_LCD_MspInit() API. After calling the HAL_LCD_Init() the LCD RAM memory is cleared
5. Optionally you can update the LCD configuration using these macros:
 - LCD High Drive using the `__HAL_LCD_HIGHDRIVER_ENABLE()` and `__HAL_LCD_HIGHDRIVER_DISABLE()` macros
 - LCD Pulse ON Duration using the `__HAL_LCD_PULSEONDURATION_CONFIG()` macro
 - LCD Dead Time using the `__HAL_LCD_DEADTIME_CONFIG()` macro
 - The LCD Blink mode and frequency using the `__HAL_LCD_BLINK_CONFIG()` macro
 - The LCD Contrast using the `__HAL_LCD_CONTRAST_CONFIG()` macro
6. Write to the LCD RAM memory using the HAL_LCD_Write() API, this API can be called more time to update the different LCD RAM registers before calling HAL_LCD_UpdateDisplayRequest() API.
7. The HAL_LCD_Clear() API can be used to clear the LCD RAM memory.

8. When LCD RAM memory is updated enable the update display request using the HAL_LCD_UpdateDisplayRequest() API.

LCD and low power modes:

1. The LCD remain active during STOP mode.

26.2.2 Initialization and Configuration functions

This section contains the following APIs:

- [*HAL_LCD_DelInit\(\)*](#)
- [*HAL_LCD_Init\(\)*](#)
- [*HAL_LCD_MspDelInit\(\)*](#)
- [*HAL_LCD_MspInit\(\)*](#)

26.2.3 IO operation functions

Using its double buffer memory the LCD controller ensures the coherency of the displayed information without having to use interrupts to control LCD_RAM modification.

- The application software can access the first buffer level (LCD_RAM) through the APB interface. Once it has modified the LCD_RAM using the HAL_LCD_Write() API, it sets the UDR flag in the LCD_SR register using the HAL_LCD_UpdateDisplayRequest() API. This UDR flag (update display request) requests the updated information to be moved into the second buffer level (LCD_DISPLAY).
- This operation is done synchronously with the frame (at the beginning of the next frame), until the update is completed, the LCD_RAM is write protected and the UDR flag stays high.
- Once the update is completed another flag (UDD - Update Display Done) is set and generates an interrupt if the UDDIE bit in the LCD_FCR register is set. The time it takes to update LCD_DISPLAY is, in the worst case, one odd and one even frame.
- The update will not occur (UDR = 1 and UDD = 0) until the display is enabled (LCDEN = 1).

This section contains the following APIs:

- [*HAL_LCD_Write\(\)*](#)
- [*HAL_LCD_Clear\(\)*](#)
- [*HAL_LCD_UpdateDisplayRequest\(\)*](#)

26.2.4 Peripheral State functions

This subsection provides a set of functions allowing to control the LCD:

- HAL_LCD_GetState() API can be helpful to check in run-time the state of the LCD peripheral State.
- HAL_LCD_GetError() API to return the LCD error code.

This section contains the following APIs:

- [*HAL_LCD_GetState\(\)*](#)
- [*HAL_LCD_GetError\(\)*](#)

26.2.5 Detailed description of functions

HAL_LCD_Delnit

Function name	HAL_StatusTypeDef HAL_LCD_Delnit (LCD_HandleTypeDef * hlcd)
Function description	DeInitializes the LCD peripheral.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_LCD_Init

Function name	HAL_StatusTypeDef HAL_LCD_Init (LCD_HandleTypeDef * hlcd)
Function description	Initializes the LCD peripheral according to the specified parameters in the LCD_InitStruct.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This function can be used only when the LCD is disabled. The LCD HighDrive can be enabled/disabled using related macros up to user.

HAL_LCD_MsplInit

Function name	void HAL_LCD_MsplInit (LCD_HandleTypeDef * hlcd)
Function description	LCD MSP Init.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_LCD_MspDelnit

Function name	void HAL_LCD_MspDelnit (LCD_HandleTypeDef * hlcd)
Function description	LCD MSP Delnit.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_LCD_Write

Function name	HAL_StatusTypeDef HAL_LCD_Write (LCD_HandleTypeDef * hlcd, uint32_t RAMRegisterIndex, uint32_t RAMRegisterMask, uint32_t Data)
Function description	Writes a word in the specific LCD RAM.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle • RAMRegisterIndex: specifies the LCD RAM Register. This parameter can be one of the following values: <ul style="list-style-type: none"> – LCD_RAM_REGISTER0: LCD RAM Register 0

	<ul style="list-style-type: none"> - LCD_RAM_REGISTER1: LCD RAM Register 1 - LCD_RAM_REGISTER2: LCD RAM Register 2 - LCD_RAM_REGISTER3: LCD RAM Register 3 - LCD_RAM_REGISTER4: LCD RAM Register 4 - LCD_RAM_REGISTER5: LCD RAM Register 5 - LCD_RAM_REGISTER6: LCD RAM Register 6 - LCD_RAM_REGISTER7: LCD RAM Register 7 - LCD_RAM_REGISTER8: LCD RAM Register 8 - LCD_RAM_REGISTER9: LCD RAM Register 9 - LCD_RAM_REGISTER10: LCD RAM Register 10 - LCD_RAM_REGISTER11: LCD RAM Register 11 - LCD_RAM_REGISTER12: LCD RAM Register 12 - LCD_RAM_REGISTER13: LCD RAM Register 13 - LCD_RAM_REGISTER14: LCD RAM Register 14 - LCD_RAM_REGISTER15: LCD RAM Register 15 <ul style="list-style-type: none"> • RAMRegisterMask: specifies the LCD RAM Register Data Mask. • Data: specifies LCD Data Value to be written.
Return values	<ul style="list-style-type: none"> • None:

HAL_LCD_Clear

Function name	HAL_StatusTypeDef HAL_LCD_Clear (LCD_HandleTypeDef * hlcd)
Function description	Clears the LCD RAM registers.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_LCD_UpdateDisplayRequest

Function name	HAL_StatusTypeDef HAL_LCD_UpdateDisplayRequest (LCD_HandleTypeDef * hlcd)
Function description	Enables the Update Display Request.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Each time software modifies the LCD_RAM it must set the UDR bit to transfer the updated data to the second level buffer. The UDR bit stays set until the end of the update and during this time the LCD_RAM is write protected. • When the display is disabled, the update is performed for all LCD_DISPLAY locations. When the display is enabled, the update is performed only for locations for which commons are active (depending on DUTY). For example if DUTY = 1/2, only the LCD_DISPLAY of COM0 and COM1 will be updated.

HAL_LCD_GetState

Function name	HAL_LCD_StateTypeDef HAL_LCD_GetState (LCD_HandleTypeDef * hlcd)
---------------	---

Function description	Returns the LCD state.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_LCD_GetError

Function name	uint32_t HAL_LCD_GetError (LCD_HandleTypeDef * hlcd)
Function description	Return the LCD error code.
Parameters	<ul style="list-style-type: none"> • hlcd: LCD handle
Return values	<ul style="list-style-type: none"> • LCD: Error Code

LCD_WaitForSynchro

Function name	HAL_StatusTypeDef LCD_WaitForSynchro (LCD_HandleTypeDef * hlcd)
Function description	Waits until the LCD FCR register is synchronized in the LCDCLK domain.
Return values	<ul style="list-style-type: none"> • None:

26.3 LCD Firmware driver defines**26.3.1 LCD*****LCD Bias***

LCD_BIAS_1_4	1/4 Bias
LCD_BIAS_1_2	1/2 Bias
LCD_BIAS_1_3	1/3 Bias
IS_LCD_BIAS	

LCD Blink Frequency

LCD_BLINKFREQUENCY_DIV8	The Blink frequency = fLCD/8
LCD_BLINKFREQUENCY_DIV16	The Blink frequency = fLCD/16
LCD_BLINKFREQUENCY_DIV32	The Blink frequency = fLCD/32
LCD_BLINKFREQUENCY_DIV64	The Blink frequency = fLCD/64
LCD_BLINKFREQUENCY_DIV128	The Blink frequency = fLCD/128
LCD_BLINKFREQUENCY_DIV256	The Blink frequency = fLCD/256
LCD_BLINKFREQUENCY_DIV512	The Blink frequency = fLCD/512
LCD_BLINKFREQUENCY_DIV1024	The Blink frequency = fLCD/1024
IS_LCD_BLINK_FREQUENCY	

LCD Blink Mode

LCD_BLINKMODE_OFF	Blink disabled
LCD_BLINKMODE_SEG0_COM0	Blink enabled on SEG[0], COM[0] (1 pixel)

LCD_BLINKMODE_SEG0_ALLCOM Blink enabled on SEG[0], all COM (up to 8 pixels according to the programmed duty)

LCD_BLINKMODE_ALLSEG_ALLCOM Blink enabled on all SEG and all COM (all pixels)

IS_LCD_BLINK_MODE

LCD Contrast

LCD_CONTRASTLEVEL_0 Maximum Voltage = 2.60V

LCD_CONTRASTLEVEL_1 Maximum Voltage = 2.73V

LCD_CONTRASTLEVEL_2 Maximum Voltage = 2.86V

LCD_CONTRASTLEVEL_3 Maximum Voltage = 2.99V

LCD_CONTRASTLEVEL_4 Maximum Voltage = 3.12V

LCD_CONTRASTLEVEL_5 Maximum Voltage = 3.25V

LCD_CONTRASTLEVEL_6 Maximum Voltage = 3.38V

LCD_CONTRASTLEVEL_7 Maximum Voltage = 3.51V

IS_LCD_CONTRAST

LCD Dead Time

LCD_DEADTIME_0 No dead Time

LCD_DEADTIME_1 One Phase between different couple of Frame

LCD_DEADTIME_2 Two Phase between different couple of Frame

LCD_DEADTIME_3 Three Phase between different couple of Frame

LCD_DEADTIME_4 Four Phase between different couple of Frame

LCD_DEADTIME_5 Five Phase between different couple of Frame

LCD_DEADTIME_6 Six Phase between different couple of Frame

LCD_DEADTIME_7 Seven Phase between different couple of Frame

IS_LCD_DEAD_TIME

LCD Divider

LCD_DIVIDER_16 LCD frequency = CLKPS/16

LCD_DIVIDER_17 LCD frequency = CLKPS/17

LCD_DIVIDER_18 LCD frequency = CLKPS/18

LCD_DIVIDER_19 LCD frequency = CLKPS/19

LCD_DIVIDER_20 LCD frequency = CLKPS/20

LCD_DIVIDER_21 LCD frequency = CLKPS/21

LCD_DIVIDER_22 LCD frequency = CLKPS/22

LCD_DIVIDER_23 LCD frequency = CLKPS/23

LCD_DIVIDER_24 LCD frequency = CLKPS/24

LCD_DIVIDER_25 LCD frequency = CLKPS/25

LCD_DIVIDER_26 LCD frequency = CLKPS/26

LCD_DIVIDER_27 LCD frequency = CLKPS/27

LCD_DIVIDER_28	LCD frequency = CLKPS/28
LCD_DIVIDER_29	LCD frequency = CLKPS/29
LCD_DIVIDER_30	LCD frequency = CLKPS/30
LCD_DIVIDER_31	LCD frequency = CLKPS/31
IS_LCD_DIVIDER	

LCD Duty

LCD_DUTY_STATIC	Static duty
LCD_DUTY_1_2	1/2 duty
LCD_DUTY_1_3	1/3 duty
LCD_DUTY_1_4	1/4 duty
LCD_DUTY_1_8	1/8 duty
IS_LCD_DUTY	

LCD Error Codes

HAL_LCD_ERROR_NONE	No error
HAL_LCD_ERROR_FCRSF	Synchro flag timeout error
HAL_LCD_ERROR_UDR	Update display request flag timeout error
HAL_LCD_ERROR_UDD	Update display done flag timeout error
HAL_LCD_ERROR_EWS	LCD enabled status flag timeout error
HAL_LCD_ERROR_RDY	LCD Booster ready timeout error

LCD Exported Macros

<u>_HAL_LCD_RESET_HANDLE_STATE</u>	Description:
	<ul style="list-style-type: none"> • Reset LCD handle state. Parameters: <ul style="list-style-type: none"> • <u>_HANDLE_</u>: specifies the LCD Handle. Return value: <ul style="list-style-type: none"> • None
<u>_HAL_LCD_ENABLE</u>	Description:
	<ul style="list-style-type: none"> • macros to enables or disables the LCD Parameters: <ul style="list-style-type: none"> • <u>_HANDLE_</u>: specifies the LCD Handle. Return value: <ul style="list-style-type: none"> • None
<u>_HAL_LCD_DISABLE</u>	
<u>_HAL_LCD_HIGHDIVER_ENABLE</u>	Description:
	<ul style="list-style-type: none"> • Macros to enable or disable the low resistance divider.

Parameters:

- __HANDLE__: specifies the LCD Handle.

Return value:

- None

Notes:

- When this mode is enabled, the PulseOn Duration (PON) have to be programmed to 1/CK_PS (LCD_PULSEONDURATION_1).

_HAL_LCD_HIGHDIVER_DISABLE
_HAL_LCD_PULSEONDURATION_C
ONFIG

Description:

- Macro to configure the LCD pulses on duration.

Parameters:

- __HANDLE__: specifies the LCD Handle.
- __DURATION__: specifies the LCD pulse on duration in terms of CK_PS (prescaled LCD clock period) pulses. This parameter can be one of the following values:
 - LCD_PULSEONDURATION_0: 0 pulse
 - LCD_PULSEONDURATION_1: Pulse ON duration = 1/CK_PS
 - LCD_PULSEONDURATION_2: Pulse ON duration = 2/CK_PS
 - LCD_PULSEONDURATION_3: Pulse ON duration = 3/CK_PS
 - LCD_PULSEONDURATION_4: Pulse ON duration = 4/CK_PS
 - LCD_PULSEONDURATION_5: Pulse ON duration = 5/CK_PS
 - LCD_PULSEONDURATION_6: Pulse ON duration = 6/CK_PS
 - LCD_PULSEONDURATION_7: Pulse ON duration = 7/CK_PS

Return value:

- None

_HAL_LCD_DEADTIME_CONFIG

Description:

- Macro to configure the LCD dead time.

Parameters:

- __HANDLE__: specifies the LCD Handle.
- __DEADTIME__: specifies the LCD dead time. This parameter can be one of the following values:
 - LCD_DEADTIME_0: No dead Time
 - LCD_DEADTIME_1: One Phase between different couple of Frame

- LCD_DEADTIME_2: Two Phase between different couple of Frame
- LCD_DEADTIME_3: Three Phase between different couple of Frame
- LCD_DEADTIME_4: Four Phase between different couple of Frame
- LCD_DEADTIME_5: Five Phase between different couple of Frame
- LCD_DEADTIME_6: Six Phase between different couple of Frame
- LCD_DEADTIME_7: Seven Phase between different couple of Frame

Return value:

- None

_HAL_LCD_CONTRAST_CONFIG**Description:**

- Macro to configure the LCD Contrast.

Parameters:

- _HANDLE_: specifies the LCD Handle.
- _CONTRAST_: specifies the LCD Contrast. This parameter can be one of the following values:
 - LCD_CONTRASTLEVEL_0: Maximum Voltage = 2.60V
 - LCD_CONTRASTLEVEL_1: Maximum Voltage = 2.73V
 - LCD_CONTRASTLEVEL_2: Maximum Voltage = 2.86V
 - LCD_CONTRASTLEVEL_3: Maximum Voltage = 2.99V
 - LCD_CONTRASTLEVEL_4: Maximum Voltage = 3.12V
 - LCD_CONTRASTLEVEL_5: Maximum Voltage = 3.25V
 - LCD_CONTRASTLEVEL_6: Maximum Voltage = 3.38V
 - LCD_CONTRASTLEVEL_7: Maximum Voltage = 3.51V

Return value:

- None

_HAL_LCD_BLINK_CONFIG**Description:**

- Macro to configure the LCD Blink mode and Blink frequency.

Parameters:

- _HANDLE_: specifies the LCD Handle.
- _BLINKMODE_: specifies the LCD blink mode. This parameter can be one of the following values:
 - LCD_BLINKMODE_OFF: Blink

- disabled
- LCD_BLINKMODE_SEG0_COM0: Blink enabled on SEG[0], COM[0] (1 pixel)
- LCD_BLINKMODE_SEG0_ALLCOM: Blink enabled on SEG[0], all COM (up to 8 pixels according to the programmed duty)
- LCD_BLINKMODE_ALLSEG_ALLCOM: Blink enabled on all SEG and all COM (all pixels)
- __BLINKFREQUENCY__: specifies the LCD blink frequency.
 - LCD_BLINKFREQUENCY_DIV8: The Blink frequency = $f_{Lcd}/8$
 - LCD_BLINKFREQUENCY_DIV16: The Blink frequency = $f_{Lcd}/16$
 - LCD_BLINKFREQUENCY_DIV32: The Blink frequency = $f_{Lcd}/32$
 - LCD_BLINKFREQUENCY_DIV64: The Blink frequency = $f_{Lcd}/64$
 - LCD_BLINKFREQUENCY_DIV128: The Blink frequency = $f_{Lcd}/128$
 - LCD_BLINKFREQUENCY_DIV256: The Blink frequency = $f_{Lcd}/256$
 - LCD_BLINKFREQUENCY_DIV512: The Blink frequency = $f_{Lcd}/512$
 - LCD_BLINKFREQUENCY_DIV1024: The Blink frequency = $f_{Lcd}/1024$

Return value:

- None

[__HAL_LCD_ENABLE_IT](#)**Description:**

- Enables or disables the specified LCD interrupt.

Parameters:

- __HANDLE__: specifies the LCD Handle.
- __INTERRUPT__: specifies the LCD interrupt source to be enabled or disabled. This parameter can be one of the following values:
 - LCD_IT_SOF: Start of Frame Interrupt
 - LCD_IT_UDD: Update Display Done Interrupt

Return value:

- None

[__HAL_LCD_DISABLE_IT](#)[__HAL_LCD_GET_IT_SOURCE](#)**Description:**

- Checks whether the specified LCD

interrupt is enabled or not.

Parameters:

- `__HANDLE__`: specifies the LCD Handle.
- `__IT__`: specifies the LCD interrupt source to check. This parameter can be one of the following values:
 - `LCD_IT_SOF`: Start of Frame Interrupt
 - `LCD_IT_UDD`: Update Display Done Interrupt.

Return value:

- The state of `__IT__` (TRUE or FALSE).

Notes:

- If the device is in STOP mode (PCLK not provided) UDD will not generate an interrupt even if UDDIE = 1. If the display is not enabled the UDD interrupt will never occur.

`__HAL_LCD_GET_FLAG`

Description:

- Checks whether the specified LCD flag is set or not.

Parameters:

- `__HANDLE__`: specifies the LCD Handle.
- `__FLAG__`: specifies the flag to check. This parameter can be one of the following values:
 - `LCD_FLAG_ENS`: LCD Enabled flag. It indicates the LCD controller status.

Return value:

- The new state of `__FLAG__` (TRUE or FALSE).

Notes:

- The ENS bit is set immediately when the LCDEN bit in the LCD_CR goes from 0 to 1. On deactivation it reflects the real status of LCD so it becomes 0 at the end of the last displayed frame. `LCD_FLAG_SOF`: Start of Frame flag. This flag is set by hardware at the beginning of a new frame, at the same time as the display data is updated. `LCD_FLAG_UDR`: Update Display Request flag. `LCD_FLAG_UDD`: Update Display Done flag. `LCD_FLAG_RDY`: Step_up converter Ready flag. It indicates the status of the step-up converter. `LCD_FLAG_FCRSF`: LCD Frame Control Register Synchronization Flag. This flag is set by

hardware each time the LCD_FCR register is updated in the LCDCLK domain.

`__HAL_LCD_CLEAR_FLAG`

Description:

- Clears the specified LCD pending flag.

Parameters:

- `__HANDLE__`: specifies the LCD Handle.
- `__FLAG__`: specifies the flag to clear. This parameter can be any combination of the following values:
 - `LCD_FLAG_SOF`: Start of Frame Interrupt
 - `LCD_FLAG_UDD`: Update Display Done Interrupt

Return value:

- None

LCD Flag

`LCD_FLAG_ENS`

`LCD_FLAG_SOF`

`LCD_FLAG_UDR`

`LCD_FLAG_UDD`

`LCD_FLAG_RDY`

`LCD_FLAG_FCRSF`

LCD HighDrive

`LCD_HIGHDRIVE_0` Low resistance Drive

`LCD_HIGHDRIVE_1` High resistance Drive

`IS_LCD_HIGHDRIVE`

LCD Interrupts

`LCD_IT_SOF`

`LCD_IT_UDD`

LCD Mux Segment

`LCD_MUXSEGMENT_DISABLE` SEG pin multiplexing disabled

`LCD_MUXSEGMENT_ENABLE` SEG[31:28] are multiplexed with SEG[43:40]

`IS_LCD_MUXSEGMENT`

LCD Prescaler

`LCD_PRESCALER_1` CLKPS = LCDCLK

`LCD_PRESCALER_2` CLKPS = LCDCLK/2

`LCD_PRESCALER_4` CLKPS = LCDCLK/4

`LCD_PRESCALER_8` CLKPS = LCDCLK/8

`LCD_PRESCALER_16` CLKPS = LCDCLK/16

LCD_PRESCALER_32	CLKPS = LCDCLK/32
LCD_PRESCALER_64	CLKPS = LCDCLK/64
LCD_PRESCALER_128	CLKPS = LCDCLK/128
LCD_PRESCALER_256	CLKPS = LCDCLK/256
LCD_PRESCALER_512	CLKPS = LCDCLK/512
LCD_PRESCALER_1024	CLKPS = LCDCLK/1024
LCD_PRESCALER_2048	CLKPS = LCDCLK/2048
LCD_PRESCALER_4096	CLKPS = LCDCLK/4096
LCD_PRESCALER_8192	CLKPS = LCDCLK/8192
LCD_PRESCALER_16384	CLKPS = LCDCLK/16384
LCD_PRESCALER_32768	CLKPS = LCDCLK/32768
IS_LCD_PRESCALER	

LCD Pulse On Duration

LCD_PULSEONDURATION_0	Pulse ON duration = 0 pulse
LCD_PULSEONDURATION_1	Pulse ON duration = 1/CK_PS
LCD_PULSEONDURATION_2	Pulse ON duration = 2/CK_PS
LCD_PULSEONDURATION_3	Pulse ON duration = 3/CK_PS
LCD_PULSEONDURATION_4	Pulse ON duration = 4/CK_PS
LCD_PULSEONDURATION_5	Pulse ON duration = 5/CK_PS
LCD_PULSEONDURATION_6	Pulse ON duration = 6/CK_PS
LCD_PULSEONDURATION_7	Pulse ON duration = 7/CK_PS
IS_LCD_PULSE_ON_DURATION	

LCD RAMRegister

LCD_RAM_REGISTER0	LCD RAM Register 0
LCD_RAM_REGISTER1	LCD RAM Register 1
LCD_RAM_REGISTER2	LCD RAM Register 2
LCD_RAM_REGISTER3	LCD RAM Register 3
LCD_RAM_REGISTER4	LCD RAM Register 4
LCD_RAM_REGISTER5	LCD RAM Register 5
LCD_RAM_REGISTER6	LCD RAM Register 6
LCD_RAM_REGISTER7	LCD RAM Register 7
LCD_RAM_REGISTER8	LCD RAM Register 8
LCD_RAM_REGISTER9	LCD RAM Register 9
LCD_RAM_REGISTER10	LCD RAM Register 10
LCD_RAM_REGISTER11	LCD RAM Register 11
LCD_RAM_REGISTER12	LCD RAM Register 12

LCD_RAM_REGISTER13 LCD RAM Register 13

LCD_RAM_REGISTER14 LCD RAM Register 14

LCD_RAM_REGISTER15 LCD RAM Register 15

IS_LCD_RAM_REGISTER

LCD Voltage Source

LCD_VOLTAGESOURCE_INTERNAL Internal voltage source for the LCD

LCD_VOLTAGESOURCE_EXTERNAL External voltage source for the LCD

IS_LCD_VOLTAGE_SOURCE

27 HAL NOR Generic Driver

27.1 NOR Firmware driver registers structures

27.1.1 NOR_IDTypeDef

Data Fields

- *uint16_t Manufacturer_Code*
- *uint16_t Device_Code1*
- *uint16_t Device_Code2*
- *uint16_t Device_Code3*

Field Documentation

- *uint16_t NOR_IDTypeDef::Manufacturer_Code*
Defines the device's manufacturer code used to identify the memory
- *uint16_t NOR_IDTypeDef::Device_Code1*
- *uint16_t NOR_IDTypeDef::Device_Code2*
- *uint16_t NOR_IDTypeDef::Device_Code3*
Defines the device's codes used to identify the memory. These codes can be accessed by performing read operations with specific control signals and addresses set. They can also be accessed by issuing an Auto Select command

27.1.2 NOR_CFITypeDef

Data Fields

- *uint16_t CFI_1*
- *uint16_t CFI_2*
- *uint16_t CFI_3*
- *uint16_t CFI_4*

Field Documentation

- *uint16_t NOR_CFITypeDef::CFI_1*
< Defines the information stored in the memory's Common flash interface which contains a description of various electrical and timing parameters, density information and functions supported by the memory
- *uint16_t NOR_CFITypeDef::CFI_2*
- *uint16_t NOR_CFITypeDef::CFI_3*
- *uint16_t NOR_CFITypeDef::CFI_4*

27.1.3 NOR_HandleTypeDef

Data Fields

- *FSMC_NORSRAM_TypeDef * Instance*
- *FSMC_NORSRAM_EXTENDED_TypeDef * Extended*
- *FSMC_NORSRAM_InitTypeDef Init*
- *HAL_LockTypeDef Lock*
- *__IO HAL_NOR_StateTypeDef State*

Field Documentation

- *FSMC_NORSRAM_TypeDef* NOR_HandleTypeDef::Instance*
Register base address

- ***FSMC_NORSRAM_EXTENDED_TypeDef* NOR_HandleTypeDef::Extended***
Extended mode register base address
- ***FSMC_NORSRAM_InitTypeDef NOR_HandleTypeDef::Init***
NOR device control configuration parameters
- ***HAL_LockTypeDef NOR_HandleTypeDef::Lock***
NOR locking object
- ***_IO HAL_NOR_StateTypeDef NOR_HandleTypeDef::State***
NOR device access state

27.2 NOR Firmware driver API description

27.2.1 How to use this driver

This driver is a generic layered driver which contains a set of APIs used to control NOR flash memories. It uses the FSMC layer functions to interface with NOR devices. This driver is used as follows:

- NOR flash memory configuration sequence using the function HAL_NOR_Init() with control and timing parameters for both normal and extended mode.
- Read NOR flash memory manufacturer code and device IDs using the function HAL_NOR_Read_ID(). The read information is stored in the NOR_ID_TypeDef structure declared by the function caller.
- Access NOR flash memory by read/write data unit operations using the functions HAL_NOR_Read(), HAL_NOR_Program().
- Perform NOR flash erase block/chip operations using the functions HAL_NOR_Erase_Block() and HAL_NOR_Erase_Chip().
- Read the NOR flash CFI (common flash interface) IDs using the function HAL_NOR_Read_CFI(). The read information is stored in the NOR_CFI_TypeDef structure declared by the function caller.
- You can also control the NOR device by calling the control APIs HAL_NOR_WriteOperation_Enable() / HAL_NOR_WriteOperation_Disable() to respectively enable/disable the NOR write operation
- You can monitor the NOR device HAL state by calling the function HAL_NOR_GetState()



This driver is a set of generic APIs which handle standard NOR flash operations. If a NOR flash device contains different operations and/or implementations, it should be implemented separately.

NOR HAL driver macros list

Below the list of most used macros in NOR HAL driver.

- NOR_WRITE : NOR memory write data to specified address

27.2.2 NOR Initialization and de_initialization functions

This section provides functions allowing to initialize/de-initialize the NOR memory

This section contains the following APIs:

- [**HAL_NOR_Init\(\)**](#)
- [**HAL_NOR_Delinit\(\)**](#)
- [**HAL_NOR_MspInit\(\)**](#)
- [**HAL_NOR_MspDelinit\(\)**](#)
- [**HAL_NOR_MspWait\(\)**](#)

27.2.3 NOR Input and Output functions

This section provides functions allowing to use and control the NOR memory

This section contains the following APIs:

- [*HAL_NOR_Read_ID\(\)*](#)
- [*HAL_NOR_ReturnToReadMode\(\)*](#)
- [*HAL_NOR_Read\(\)*](#)
- [*HAL_NOR_Program\(\)*](#)
- [*HAL_NOR_ReadBuffer\(\)*](#)
- [*HAL_NOR_ProgramBuffer\(\)*](#)
- [*HAL_NOR_Erase_Block\(\)*](#)
- [*HAL_NOR_Erase_Chip\(\)*](#)
- [*HAL_NOR_Read_CFI\(\)*](#)

27.2.4 NOR Control functions

This subsection provides a set of functions allowing to control dynamically the NOR interface.

This section contains the following APIs:

- [*HAL_NOR_WriteOperation_Enable\(\)*](#)
- [*HAL_NOR_WriteOperation_Disable\(\)*](#)

27.2.5 NOR State functions

This subsection permits to get in run-time the status of the NOR controller and the data flow.

This section contains the following APIs:

- [*HAL_NOR_GetState\(\)*](#)
- [*HAL_NOR_GetStatus\(\)*](#)

27.2.6 Detailed description of functions

HAL_NOR_Init

Function name **HAL_StatusTypeDef HAL_NOR_Init (NOR_HandleTypeDef *
hnor, FSMC_NORSRAM_TimingTypeDef * Timing,
FSMC_NORSRAM_TimingTypeDef * ExtTiming)**

Function description Perform the NOR memory Initialization sequence.

Parameters • **hnor**: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.
• **Timing**: pointer to NOR control timing structure
• **ExtTiming**: pointer to NOR extended mode timing structure

Return values • **HAL**: status

HAL_NOR_DeInit

Function name **HAL_StatusTypeDef HAL_NOR_DeInit (NOR_HandleTypeDef *
hnor)**

Function description Perform NOR memory De-Initialization sequence.

Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_MspInit

Function name	void HAL_NOR_MspInit (NOR_HandleTypeDef * hnor)
Function description	NOR MSP Init.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.
Return values	<ul style="list-style-type: none"> • None:

HAL_NOR_MspDelInit

Function name	void HAL_NOR_MspDelInit (NOR_HandleTypeDef * hnor)
Function description	NOR MSP DelInit.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.
Return values	<ul style="list-style-type: none"> • None:

HAL_NOR_MspWait

Function name	void HAL_NOR_MspWait (NOR_HandleTypeDef * hnor, uint32_t Timeout)
Function description	NOR MSP Wait fro Ready/Busy signal.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • Timeout: Maximum timeout value
Return values	<ul style="list-style-type: none"> • None:

HAL_NOR_Read_ID

Function name	HAL_StatusTypeDef HAL_NOR_Read_ID (NOR_HandleTypeDef * hnor, NOR_IDTypeDef * pNOR_ID)
Function description	Read NOR flash IDs.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • pNOR_ID: : pointer to NOR ID structure
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_ReturnToReadMode

Function name	HAL_StatusTypeDef HAL_NOR_ReturnToReadMode (NOR_HandleTypeDef * hnor)
Function description	Returns the NOR memory to Read mode.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that

contains the configuration information for NOR module.

Return values	<ul style="list-style-type: none"> • HAL: status
---------------	--

HAL_NOR_Read

Function name	HAL_StatusTypeDef HAL_NOR_Read (NOR_HandleTypeDef *hnor, uint32_t * pAddress, uint16_t * pData)
Function description	Read data from NOR memory.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • pAddress: pointer to Device address • pData: : pointer to read data
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_Program

Function name	HAL_StatusTypeDef HAL_NOR_Program (NOR_HandleTypeDef * hnор, uint32_t * pAddress, uint16_t * pData)
Function description	Program data to NOR memory.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • pAddress: Device address • pData: : pointer to the data to write
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_ReadBuffer

Function name	HAL_StatusTypeDef HAL_NOR_ReadBuffer (NOR_HandleTypeDef * hnор, uint32_t uwAddress, uint16_t * pData, uint32_t uwBufferSize)
Function description	Reads a block of data from the FSMC NOR memory.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • uwAddress: NOR memory internal address to read from. • pData: pointer to the buffer that receives the data read from the NOR memory. • uwBufferSize: : number of Half word to read.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_ProgramBuffer

Function name	HAL_StatusTypeDef HAL_NOR_ProgramBuffer (NOR_HandleTypeDef * hnор, uint32_t uwAddress, uint16_t * pData, uint32_t uwBufferSize)
Function description	Writes a half-word buffer to the FSMC NOR memory.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.

	<ul style="list-style-type: none"> • uwAddress: NOR memory internal address from which the data • pData: pointer to source data buffer. • uwBufferSize: number of Half words to write.
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • Some NOR memory need Address aligned to xx bytes (can be aligned to 64 bytes boundary for example). • The maximum buffer size allowed is NOR memory dependent (can be 64 Bytes max for example).

HAL_NOR_Erase_Block

Function name	HAL_StatusTypeDef HAL_NOR_Erase_Block (NOR_HandleTypeDef * hnor, uint32_t BlockAddress, uint32_t Address)
Function description	Erase the specified block of the NOR memory.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • BlockAddress: : Block to erase address • Address: Device address
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_Erase_Chip

Function name	HAL_StatusTypeDef HAL_NOR_Erase_Chip (NOR_HandleTypeDef * hnor, uint32_t Address)
Function description	Erase the entire NOR chip.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • Address: : Device address
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_Read_CFI

Function name	HAL_StatusTypeDef HAL_NOR_Read_CFI (NOR_HandleTypeDef * hnor, NOR_CFITypeDef * pNOR_CFI)
Function description	Read NOR flash CFI IDs.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • pNOR_CFI: : pointer to NOR CFI IDs structure
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_WriteOperation_Enable

Function name	HAL_StatusTypeDef HAL_NOR_WriteOperation_Enable (NOR_HandleTypeDef * hnor)
Function description	Enables dynamically NOR write operation.

Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_WriteOperation_Disable

Function name	HAL_StatusTypeDef HAL_NOR_WriteOperation_Disable (NOR_HandleTypeDef * hnor)
Function description	Disables dynamically NOR write operation.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_NOR_GetState

Function name	HAL_NOR_StateTypeDef HAL_NOR_GetState (NOR_HandleTypeDef * hnor)
Function description	return the NOR controller state
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.
Return values	<ul style="list-style-type: none"> • NOR: controller state

HAL_NOR_GetStatus

Function name	HAL_NOR_StatusTypeDef HAL_NOR_GetStatus (NOR_HandleTypeDef * hnor, uint32_t Address, uint32_t Timeout)
Function description	Returns the NOR operation status.
Parameters	<ul style="list-style-type: none"> • hnor: pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module. • Address: Device address • Timeout: NOR programming Timeout
Return values	<ul style="list-style-type: none"> • NOR_Status: The returned value can be: HAL_NOR_STATUS_SUCCESS, HAL_NOR_STATUS_ERROR or HAL_NOR_STATUS_TIMEOUT

27.3 NOR Firmware driver defines

27.3.1 NOR

NOR Exported Macros

<u>_HAL_NOR_RESET_HANDLE_STATE</u>	Description:
	<ul style="list-style-type: none"> • Reset NOR handle state.
	Parameters: <ul style="list-style-type: none"> • <u>_HANDLE_</u>: NOR handle

Return value:

- None

28 HAL OPAMP Generic Driver

28.1 OPAMP Firmware driver registers structures

28.1.1 OPAMP_InitTypeDef

Data Fields

- *uint32_t PowerSupplyRange*
- *uint32_t PowerMode*
- *uint32_t Mode*
- *uint32_t InvertingInput*
- *uint32_t NonInvertingInput*
- *uint32_t UserTrimming*
- *uint32_t TrimmingValueP*
- *uint32_t TrimmingValueN*
- *uint32_t TrimmingValuePLowPower*
- *uint32_t TrimmingValueNLowPower*

Field Documentation

- ***uint32_t OPAMP_InitTypeDef::PowerSupplyRange***
Specifies the power supply range: above or under 2.4V. This parameter must be a value of **OPAMP_PowerSupplyRange** Caution: This parameter is common to all OPAMP instances: a modification of this parameter for the selected OPAMP impacts the other OPAMP instances.
- ***uint32_t OPAMP_InitTypeDef::PowerMode***
Specifies the power mode Normal or Low-Power. This parameter must be a value of **OPAMP_PowerMode**
- ***uint32_t OPAMP_InitTypeDef::Mode***
Specifies the OPAMP mode This parameter must be a value of **OPAMP_Mode** mode is either Standalone or Follower
- ***uint32_t OPAMP_InitTypeDef::InvertingInput***
Specifies the inverting input in Standalone modeIn Standalone mode: i.e when mode is OPAMP_STANDALONE_MODE This parameter must be a value of **OPAMP_InvertingInput** InvertingInput is either VM0 or VM1In Follower mode: i.e when mode is OPAMP_FOLLOWER_MODE This parameter is Not Applicable
- ***uint32_t OPAMP_InitTypeDef::NonInvertingInput***
Specifies the non inverting input of the opamp: This parameter must be a value of **OPAMP_NonInvertingInput** Note: Non-inverting input availability depends on OPAMP instance: OPAMP1: Non-inverting input is either IO0, DAC_Channel1
OPAMP2: Non-inverting input is either IO0, DAC_Channel1, DAC_Channel2
OPAMP3: Non-inverting input is either IO0, DAC_Channel2 (OPAMP3 availability depends on STM32L1 devices)
- ***uint32_t OPAMP_InitTypeDef::UserTrimming***
Specifies the trimming mode This parameter must be a value of **OPAMP_UserTrimming** UserTrimming is either factory or user trimming. Caution: This parameter is common to all OPAMP instances: a modification of this parameter for the selected OPAMP impacts the other OPAMP instances.
- ***uint32_t OPAMP_InitTypeDef::TrimmingValueP***
Specifies the offset trimming value (PMOS) i.e. when UserTrimming is OPAMP_TRIMMING_USER. This parameter must be a number between Min_Data = 0 and Max_Data = 30 (Trimming value 31 is forbidden) 16 is typical default value

- ***uint32_t OPAMP_InitTypeDef::TrimmingValueN***
Specifies the offset trimming value (NMOS) i.e. when UserTrimming is OPAMP_TRIMMING_USER. This parameter must be a number between Min_Data = 0 and Max_Data = 30 (Trimming value 31 is forbidden) 16 is typical default value
- ***uint32_t OPAMP_InitTypeDef::TrimmingValuePLowPower***
Specifies the offset trimming value (PMOS) i.e. when UserTrimming is OPAMP_TRIMMING_USER. This parameter must be a number between Min_Data = 0 and Max_Data = 30 (Trimming value 31 is forbidden) 16 is typical default value
- ***uint32_t OPAMP_InitTypeDef::TrimmingValueNLowPower***
Specifies the offset trimming value (NMOS) i.e. when UserTrimming is OPAMP_TRIMMING_USER. This parameter must be a number between Min_Data = 0 and Max_Data = 30 (Trimming value 31 is forbidden) 16 is typical default value

28.1.2 OPAMP_HandleTypeDef

Data Fields

- ***OPAMP_TypeDef * Instance***
- ***OPAMP_InitTypeDef Init***
- ***HAL_StatusTypeDef Status***
- ***HAL_LockTypeDef Lock***
- ***__IO HAL_OPAMP_StateTypeDef State***

Field Documentation

- ***OPAMP_TypeDef* OPAMP_HandleTypeDef::Instance***
OPAMP instance's registers base address
- ***OPAMP_InitTypeDef OPAMP_HandleTypeDef::Init***
OPAMP required parameters
- ***HAL_StatusTypeDef OPAMP_HandleTypeDef::Status***
OPAMP peripheral status
- ***HAL_LockTypeDef OPAMP_HandleTypeDef::Lock***
Locking object
- ***__IO HAL_OPAMP_StateTypeDef OPAMP_HandleTypeDef::State***
OPAMP communication state

28.2 OPAMP Firmware driver API description

28.2.1 OPAMP Peripheral Features

The device integrates up to 3 operational amplifiers OPAMP1, OPAMP2, OPAMP3 (OPAMP3 availability depends on device category)

1. The OPAMP(s) provides several exclusive running modes.
 - Standalone mode
 - Follower mode
2. All OPAMP (same for all OPAMPs) can operate in
 - Either Low range ($VDDA < 2.4V$) power supply
 - Or High range ($VDDA > 2.4V$) power supply
3. Each OPAMP(s) can be configured in normal and low power mode.
4. The OPAMP(s) provide(s) calibration capabilities.
 - Calibration aims at correcting some offset for running mode.
 - The OPAMP uses either factory calibration settings OR user defined calibration (trimming) settings (i.e. trimming mode).
 - The user defined settings can be figured out using self calibration handled by HAL_OPAMP_SelfCalibrate, HAL_OPAMPEx_SelfCalibrateAll
 - HAL_OPAMP_SelfCalibrate:

- Runs automatically the calibration in 2 steps: for transistors differential pair high (PMOS) or low (NMOS)
 - Enables the user trimming mode
 - Updates the init structure with trimming values with fresh calibration results. The user may store the calibration results for larger (ex monitoring the trimming as a function of temperature for instance)
 - For devices having several OPAMPs, HAL_OPAMPEx_SelfCalibrateAll runs calibration of all OPAMPs in parallel to save search time.
5. Running mode: Standalone mode
 - Gain is set externally (gain depends on external loads).
 - Follower mode also possible externally by connecting the inverting input to the output.
 6. Running mode: Follower mode
 - No Inverting Input is connected.
 - The OPAMP(s) output(s) are internally connected to inverting input.

28.2.2 How to use this driver

power supply range

To run in low power mode:

1. Configure the opamp using HAL_OPAMP_Init() function:
 - Select OPAMP_SUPPLY_LOW (VDDA lower than 2.4V)
 - Otherwise select OPAMP_SUPPLY_HIGH (VDDA higher than 2.4V)

low / normal power mode

To run in low power mode:

1. Configure the opamp using HAL_OPAMP_Init() function:
 - Select OPAMP_POWERMODE_LOWPOWER
 - Otherwise select OPAMP_POWERMODE_NORMAL

Calibration

To run the opamp calibration self calibration:

1. Start calibration using HAL_OPAMP_SelfCalibrate. Store the calibration results.

Running mode

To use the opamp, perform the following steps:

1. Fill in the HAL_OPAMP_MspInit() to
 - Enable the OPAMP Peripheral clock using macro
"__HAL_RCC_OPAMP_CLK_ENABLE()"
 - Configure the opamp input AND output in analog mode using HAL_GPIO_Init() to map the opamp output to the GPIO pin.
2. Configure the opamp using HAL_OPAMP_Init() function:
 - Select the mode
 - Select the inverting input
 - Select the non-inverting input
 - Select either factory or user defined trimming mode.
 - If the user defined trimming mode is enabled, select PMOS & NMOS trimming values (typ. settings returned by HAL_OPAMP_SelfCalibrate function).

3. Enable the opamp using HAL_OPAMP_Start() function.
4. Disable the opamp using HAL_OPAMP_Stop() function.
5. Lock the opamp in running mode using HAL_OPAMP_Lock() function. Caution: On STM32L1, HAL OPAMP lock is software lock only (not hardware lock as on some other STM32 devices)
6. If needed, unlock the opamp using HAL_OPAMPEx_Unlock() function.

Running mode: change of configuration while OPAMP ON

To Re-configure OPAMP when OPAMP is ON (change on the fly)

1. If needed, Fill in the HAL_OPAMP_MspInit()
 - This is the case for instance if you wish to use new OPAMP I/O
2. Configure the opamp using HAL_OPAMP_Init() function:
 - As in configure case, selects first the parameters you wish to modify.
3. Change from low power mode to normal power mode (& vice versa) requires first HAL_OPAMP_DeInit() (force OPAMP OFF) and then HAL_OPAMP_Init(). In other words, of OPAMP is ON, HAL_OPAMP_Init can NOT change power mode alone.

28.2.3 Initialization and de-initialization functions

This section provides functions allowing to:

This section contains the following APIs:

- [**HAL_OPAMP_Init\(\)**](#)
- [**HAL_OPAMP_DeInit\(\)**](#)
- [**HAL_OPAMP_MspInit\(\)**](#)
- [**HAL_OPAMP_MspDeInit\(\)**](#)

28.2.4 IO operation functions

This subsection provides a set of functions allowing to manage the OPAMP start, stop and calibration actions.

This section contains the following APIs:

- [**HAL_OPAMP_Start\(\)**](#)
- [**HAL_OPAMP_Stop\(\)**](#)
- [**HAL_OPAMP_SelfCalibrate\(\)**](#)

28.2.5 Peripheral Control functions

This subsection provides a set of functions allowing to control the OPAMP data transfers.

This section contains the following APIs:

- [**HAL_OPAMP_Lock\(\)**](#)
- [**HAL_OPAMP_GetTrimOffset\(\)**](#)

28.2.6 Peripheral State functions

This subsection permit to get in run-time the status of the peripheral.

This section contains the following APIs:

- [**HAL_OPAMP_GetState\(\)**](#)

28.2.7 Detailed description of functions

HAL_OPAMP_Init

Function name	HAL_StatusTypeDef HAL_OPAMP_Init (OPAMP_HandleTypeDef * hopamp)
Function description	Initializes the OPAMP according to the specified parameters in the OPAMP_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • hopamp: OPAMP handle
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • If the selected opamp is locked, initialization can't be performed. To unlock the configuration, perform a system reset.

HAL_OPAMP_DelInit

Function name	HAL_StatusTypeDef HAL_OPAMP_DelInit (OPAMP_HandleTypeDef * hopamp)
Function description	Deinitializes the OPAMP peripheral.
Parameters	<ul style="list-style-type: none"> • hopamp: OPAMP handle
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • Deinitialization can be performed if the OPAMP configuration is locked. (the OPAMP lock is SW in STM32L1)

HAL_OPAMP_MsplInit

Function name	void HAL_OPAMP_MsplInit (OPAMP_HandleTypeDef * hopamp)
Function description	Initializes the OPAMP MSP.
Parameters	<ul style="list-style-type: none"> • hopamp: OPAMP handle
Return values	<ul style="list-style-type: none"> • None:

HAL_OPAMP_MspDelInit

Function name	void HAL_OPAMP_MspDelInit (OPAMP_HandleTypeDef * hopamp)
Function description	Deinitializes OPAMP MSP.
Parameters	<ul style="list-style-type: none"> • hopamp: OPAMP handle
Return values	<ul style="list-style-type: none"> • None:

HAL_OPAMP_Start

Function name	HAL_StatusTypeDef HAL_OPAMP_Start (OPAMP_HandleTypeDef * hopamp)
Function description	Start the opamp.

Parameters	<ul style="list-style-type: none"> hopamp: OPAMP handle
Return values	<ul style="list-style-type: none"> HAL: status

HAL_OPAMP_Stop

Function name	HAL_StatusTypeDef HAL_OPAMP_Stop (OPAMP_HandleTypeDef * hopamp)
Function description	Stop the opamp.
Parameters	<ul style="list-style-type: none"> hopamp: OPAMP handle
Return values	<ul style="list-style-type: none"> HAL: status

HAL_OPAMP_SelfCalibrate

Function name	HAL_StatusTypeDef HAL_OPAMP_SelfCalibrate (OPAMP_HandleTypeDef * hopamp)
Function description	Run the self calibration of one OPAMP.
Parameters	<ul style="list-style-type: none"> hopamp: handle
Return values	<ul style="list-style-type: none"> Updated: offset trimming values (PMOS & NMOS), user trimming is enabled HAL: status
Notes	<ul style="list-style-type: none"> Trimming values (PMOS & NMOS) are updated and user trimming is enabled if calibration is successful. Calibration is performed in the mode specified in OPAMP init structure (mode normal or low-power). To perform calibration for both modes, repeat this function twice after OPAMP init structure accordingly updated. Calibration runs about 10 ms.

HAL_OPAMP_Lock

Function name	HAL_StatusTypeDef HAL_OPAMP_Lock (OPAMP_HandleTypeDef * hopamp)
Function description	Lock the selected opamp configuration.
Parameters	<ul style="list-style-type: none"> hopamp: OPAMP handle
Return values	<ul style="list-style-type: none"> HAL: status

HAL_OPAMP_GetTrimOffset

Function name	HAL_OPAMP_TrimmingValueTypeDef HAL_OPAMP_GetTrimOffset (OPAMP_HandleTypeDef * hopamp, uint32_t trimmingoffset)
Function description	Return the OPAMP factory trimming value Caution: On STM32L1 OPAMP, user can retrieve factory trimming if OPAMP has never been set to user trimming before.
Parameters	<ul style="list-style-type: none"> hopamp: : OPAMP handle trimmingoffset: : Trimming offset (P or N) This parameter must be a value of OPAMP FactoryTrimming

- | | |
|---------------|--|
| Return values | <ul style="list-style-type: none"> • Trimming: value (P or N): range: 0->31 or OPAMP_FACTORYTRIMMING_DUMMY if trimming value is not available |
| Notes | <ul style="list-style-type: none"> • Calibration parameter retrieved is corresponding to the mode specified in OPAMP init structure (mode normal or low-power). To retrieve calibration parameters for both modes, repeat this function after OPAMP init structure accordingly updated. |

HAL_OPAMP_GetState

Function name	HAL_OPAMP_StateTypeDef HAL_OPAMP_GetState (OPAMP_HandleTypeDef * hopamp)
Function description	Return the OPAMP state.
Parameters	<ul style="list-style-type: none"> • hopamp: : OPAMP handle
Return values	<ul style="list-style-type: none"> • HAL: state

28.3 OPAMP Firmware driver defines

28.3.1 OPAMP

OPAMP Exported Constants

OPAMP_TRIM_VALUE_MASK

OPAMP_CSR_INSTANCE_OFFSET

OPAMP_OTR_INSTANCE_OFFSET

OPAMP FactoryTrimming

OPAMP_FACTORYTRIMMING_DUMMY Dummy value if trimming value could not be retrieved

OPAMP_FACTORYTRIMMING_P Offset trimming P

OPAMP_FACTORYTRIMMING_N Offset trimming N

OPAMP InvertingInput

OPAMP_INVERTINGINPUT_IO0 Comparator inverting input connected to dedicated IO pin low-leakage

OPAMP_INVERTINGINPUT_IO1 Comparator inverting input connected to alternative IO pin available on some device packages

OPAMP Mode

OPAMP_STANDALONE_MODE OPAMP standalone mode

OPAMP_FOLLOWER_MODE OPAMP follower mode

OPAMP NonInvertingInput

OPAMP_NONINVERTINGINPUT_IO0 Comparator non-inverting input connected to dedicated IO pin low-leakage

OPAMP_NONINVERTINGINPUT_DAC_CH1 Comparator non-inverting input connected internally to DAC channel 1. Available only on OPAMP1 and OPAMP2.

OPAMP_NONINVERTINGINPUT_DAC_CH2 Comparator non-inverting input connected internally to DAC channel 2. Available only on OPAMP2 and OPAMP3 (OPAMP3 availability depends on STM32L1 devices).

OPAMP PowerMode

OPAMP_POWERMODE_NORMAL

OPAMP_POWERMODE_LOWPOWER

OPAMP PowerSupplyRange

OPAMP_POWERSUPPLY_LOW Power supply range low (VDDA lower than 2.4V)

OPAMP_POWERSUPPLY_HIGH Power supply range high (VDDA higher than 2.4V)

OPAMP User Trimming

OPAMP_TRIMMING_FACTORY Factory trimming

OPAMP_TRIMMING_USER User trimming

29 HAL OPAMP Extension Driver

29.1 OPAMPEx Firmware driver API description

29.1.1 Peripheral Control functions

- OPAMP unlock.

This section contains the following APIs:

- [*HAL_OPAMPEx_Unlock\(\)*](#)

29.1.2 Extended IO operation functions

- OPAMP Self calibration.

This section contains the following APIs:

- [*HAL_OPAMPEx_SelfCalibrateAll\(\)*](#)

29.1.3 Detailed description of functions

HAL_OPAMPEx_SelfCalibrateAll

Function name	HAL_StatusTypeDef HAL_OPAMPEx_SelfCalibrateAll (OPAMP_HandleTypeDef * hopamp1, OPAMP_HandleTypeDef * hopamp2, OPAMP_HandleTypeDef * hopamp3)
Function description	Run the self calibration of the 3 OPAMPs in parallel.
Parameters	<ul style="list-style-type: none"> • hopamp1: handle • hopamp2: handle • hopamp3: handle
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • Trimming values (PMOS & NMOS) are updated and user trimming is enabled is calibration is successful. • Calibration is performed in the mode specified in OPAMP init structure (mode normal or low-power). To perform calibration for both modes, repeat this function twice after OPAMP init structure accordingly updated. • Calibration runs about 10 ms (5 dichotomy steps, repeated for P and N transistors: 10 steps with 1 ms for each step).

HAL_OPAMPEx_Unlock

Function name	HAL_StatusTypeDef HAL_OPAMPEx_Unlock (OPAMP_HandleTypeDef * hopamp)
Function description	Unlock the selected opamp configuration.
Parameters	<ul style="list-style-type: none"> • hopamp: OPAMP handle
Return values	<ul style="list-style-type: none"> • HAL: status

29.2 OPAMPEx Firmware driver defines

29.2.1 OPAMPEx

OPAMPEx Exported Constants

OPAMP_CSR_OPAXPD_ALL
OPAMP_CSR_OPAXCAL_L_ALL
OPAMP_CSR_OPAXCAL_H_ALL
OPAMP_CSR_ALL_SWITCHES_ALL_OPAMPS

OPAMPEx Exported Macro

`_HAL_OPAMP_OPAMP3OUT_CONNECT_ADC_COMP1`

Description:

- Enable internal analog switch SW1 to connect OPAMP3 output to ADC switch matrix (ADC channel VCOMP, channel 26) and COMP1 non-inverting input (OPAMP3 available on STM32L1 devices Cat.4 only).

Return value:

- None

`_HAL_OPAMP_OPAMP3OUT_DISCONNECT_ADC_COMP1`

Description:

- Disable internal analog switch SW1 to disconnect OPAMP3 output from ADC switch matrix (ADC channel VCOMP, channel 26) and COMP1 non-inverting input.

Return value:

- None

30 HAL PCD Generic Driver

30.1 PCD Firmware driver registers structures

30.1.1 PCD_InitTypeDef

Data Fields

- *uint32_t dev_endpoints*
- *uint32_t speed*
- *uint32_t ep0_mps*
- *uint32_t phy_itface*
- *uint32_t Sof_enable*
- *uint32_t low_power_enable*
- *uint32_t lpm_enable*
- *uint32_t battery_charging_enable*

Field Documentation

- ***uint32_t PCD_InitTypeDef::dev_endpoints***
Device Endpoints number. This parameter depends on the used USB core. This parameter must be a number between Min_Data = 1 and Max_Data = 15
- ***uint32_t PCD_InitTypeDef::speed***
USB Core speed. This parameter can be any value of [PCD_Core_Speed](#)
- ***uint32_t PCD_InitTypeDef::ep0_mps***
Set the Endpoint 0 Max Packet size. This parameter can be any value of [PCD_EP0_MPS](#)
- ***uint32_t PCD_InitTypeDef::phy_itface***
Select the used PHY interface. This parameter can be any value of [PCD_Core_PHY](#)
- ***uint32_t PCD_InitTypeDef::Sof_enable***
Enable or disable the output of the SOF signal. This parameter can be set to ENABLE or DISABLE
- ***uint32_t PCD_InitTypeDef::low_power_enable***
Enable or disable Low Power mode This parameter can be set to ENABLE or DISABLE
- ***uint32_t PCD_InitTypeDef::lpm_enable***
Enable or disable the Link Power Management . This parameter can be set to ENABLE or DISABLE
- ***uint32_t PCD_InitTypeDef::battery_charging_enable***
Enable or disable Battery charging. This parameter can be set to ENABLE or DISABLE

30.1.2 PCD_EPTypeDef

Data Fields

- *uint8_t num*
- *uint8_t is_in*
- *uint8_t is_stall*
- *uint8_t type*
- *uint16_t pmaaddress*
- *uint16_t pmaaddr0*
- *uint16_t pmaaddr1*
- *uint8_t doublebuffer*

- *uint32_t maxpacket*
- *uint8_t * xfer_buff*
- *uint32_t xfer_len*
- *uint32_t xfer_count*

Field Documentation

- ***uint8_t PCD_EPTypedef::num***
Endpoint number This parameter must be a number between Min_Data = 1 and Max_Data = 15
- ***uint8_t PCD_EPTypedef::is_in***
Endpoint direction This parameter must be a number between Min_Data = 0 and Max_Data = 1
- ***uint8_t PCD_EPTypedef::is_stall***
Endpoint stall condition This parameter must be a number between Min_Data = 0 and Max_Data = 1
- ***uint8_t PCD_EPTypedef::type***
Endpoint type This parameter can be any value of [PCD_EP_Type](#)
- ***uint16_t PCD_EPTypedef::pmaaddress***
PMA Address This parameter can be any value between Min_addr = 0 and Max_addr = 1K
- ***uint16_t PCD_EPTypedef::pmaaddr0***
PMA Address0 This parameter can be any value between Min_addr = 0 and Max_addr = 1K
- ***uint16_t PCD_EPTypedef::pmaaddr1***
PMA Address1 This parameter can be any value between Min_addr = 0 and Max_addr = 1K
- ***uint8_t PCD_EPTypedef::doublebuffer***
Double buffer enable This parameter can be 0 or 1
- ***uint32_t PCD_EPTypedef::maxpacket***
Endpoint Max packet size This parameter must be a number between Min_Data = 0 and Max_Data = 64KB
- ***uint8_t* PCD_EPTypedef::xfer_buff***
Pointer to transfer buffer
- ***uint32_t PCD_EPTypedef::xfer_len***
Current transfer length
- ***uint32_t PCD_EPTypedef::xfer_count***
Partial transfer length in case of multi packet transfer

30.1.3 PCD_HandleTypeDef

Data Fields

- *PCD_TypeDef * Instance*
- *PCD_InitTypeDef Init*
- *_IO uint8_t USB_Address*
- *PCD_EPTypedef IN_ep*
- *PCD_EPTypedef OUT_ep*
- *HAL_LockTypeDef Lock*
- *_IO PCD_StateTypeDef State*
- *uint32_t Setup*
- *void * pData*

Field Documentation

- ***PCD_TypeDef* PCD_HandleTypeDef::Instance***
Register base address

- ***PCD_InitTypeDef PCD_HandleTypeDef::Init***
PCD required parameters
- ***_IO uint8_t PCD_HandleTypeDef::USB_Address***
USB Address
- ***PCD_EPTypeDef PCD_HandleTypeDef::IN_ep[8]***
IN endpoint parameters
- ***PCD_EPTypeDef PCD_HandleTypeDef::OUT_ep[8]***
OUT endpoint parameters
- ***HAL_LockTypeDef PCD_HandleTypeDef::Lock***
PCD peripheral status
- ***_IO PCD_StateTypeDef PCD_HandleTypeDef::State***
PCD communication state
- ***uint32_t PCD_HandleTypeDef::Setup[12]***
Setup packet buffer
- ***void* PCD_HandleTypeDef::pData***
Pointer to upper stack Handler

30.2 PCD Firmware driver API description

30.2.1 How to use this driver

The PCD HAL driver can be used as follows:

1. Declare a PCD_HandleTypeDef handle structure, for example: PCD_HandleTypeDef hpcd;
2. Fill parameters of Init structure in HCD handle
3. Call HAL_PCD_Init() API to initialize the HCD peripheral (Core, Device core, ...)
4. Initialize the PCD low level resources through the HAL_PCD_MspInit() API:
 - a. Enable the PCD/USB Low Level interface clock using
 - __HAL_RCC_USB_CLK_ENABLE();
 - b. Initialize the related GPIO clocks
 - c. Configure PCD pin-out
 - d. Configure PCD NVIC interrupt
5. Associate the Upper USB device stack to the HAL PCD Driver:
 - a. hpcd.pData = pdev;
6. Enable HCD transmission and reception:
 - a. HAL_PCD_Start();

30.2.2 Initialization and de-initialization functions

This section provides functions allowing to:

This section contains the following APIs:

- [**HAL_PCD_Init\(\)**](#)
- [**HAL_PCD_DelInit\(\)**](#)
- [**HAL_PCD_MspInit\(\)**](#)
- [**HAL_PCD_MspDelInit\(\)**](#)

30.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the PCD data transfers.

This section contains the following APIs:

- [**HAL_PCD_Start\(\)**](#)
- [**HAL_PCD_Stop\(\)**](#)
- [**HAL_PCD_IRQHandler\(\)**](#)

- `HAL_PCD_DataOutStageCallback()`
- `HAL_PCD_DataInStageCallback()`
- `HAL_PCD_SetupStageCallback()`
- `HAL_PCD_SOFCallback()`
- `HAL_PCD_ResetCallback()`
- `HAL_PCD_SuspendCallback()`
- `HAL_PCD_ResumeCallback()`
- `HAL_PCD_ISOOUTIncompleteCallback()`
- `HAL_PCD_ISOINIncompleteCallback()`
- `HAL_PCD_ConnectCallback()`
- `HAL_PCD_DisconnectCallback()`

30.2.4 Peripheral Control functions

This subsection provides a set of functions allowing to control the PCD data transfers.

This section contains the following APIs:

- `HAL_PCD_DevConnect()`
- `HAL_PCD_DevDisconnect()`
- `HAL_PCD_SetAddress()`
- `HAL_PCD_EP_Open()`
- `HAL_PCD_EP_Close()`
- `HAL_PCD_EP_Receive()`
- `HAL_PCD_EP_GetRxCount()`
- `HAL_PCD_EP_Transmit()`
- `HAL_PCD_EP_SetStall()`
- `HAL_PCD_EP_ClrStall()`
- `HAL_PCD_EP_Flush()`
- `HAL_PCD_ActivateRemoteWakeup()`
- `HAL_PCD_DeActivateRemoteWakeup()`

30.2.5 Peripheral State functions

This subsection permits to get in run-time the status of the peripheral and the data flow.

This section contains the following APIs:

- `HAL_PCD_GetState()`
- `HAL_PCDEx_SetConnectionState()`

30.2.6 Detailed description of functions

`HAL_PCD_Init`

Function name	<code>HAL_StatusTypeDef HAL_PCD_Init (PCD_HandleTypeDef * hpcd)</code>
Function description	Initializes the PCD according to the specified parameters in the PCD_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • <code>hpcd</code>: PCD handle
Return values	<ul style="list-style-type: none"> • <code>HAL</code>: status

`HAL_PCD_DelInit`

Function name	<code>HAL_StatusTypeDef HAL_PCD_DelInit (PCD_HandleTypeDef * hpcd)</code>
---------------	---

hpcd)

Function description Deinitializes the PCD peripheral.

Parameters • **hpcd:** PCD handle

Return values • **HAL:** status

HAL_PCD_MsplInit

Function name **void HAL_PCD_MsplInit (PCD_HandleTypeDef * hpcd)**

Function description Initializes the PCD MSP.

Parameters • **hpcd:** PCD handle

Return values • **None:**

HAL_PCD_MspDeInit

Function name **void HAL_PCD_MspDeInit (PCD_HandleTypeDef * hpcd)**

Function description Deinitializes PCD MSP.

Parameters • **hpcd:** PCD handle

Return values • **None:**

HAL_PCD_Start

Function name **HAL_StatusTypeDef HAL_PCD_Start (PCD_HandleTypeDef * hpcd)**

Function description Start the USB device.

Parameters • **hpcd:** PCD handle

Return values • **HAL:** status

HAL_PCD_Stop

Function name **HAL_StatusTypeDef HAL_PCD_Stop (PCD_HandleTypeDef * hpcd)**

Function description Stop the USB device.

Parameters • **hpcd:** PCD handle

Return values • **HAL:** status

HAL_PCD_IRQHandler

Function name **void HAL_PCD_IRQHandler (PCD_HandleTypeDef * hpcd)**

Function description This function handles PCD interrupt request.

Parameters • **hpcd:** PCD handle

Return values • **HAL:** status

HAL_PCD_DataOutStageCallback

Function name	void HAL_PCD_DataOutStageCallback (PCD_HandleTypeDef * hpcd, uint8_t epnum)
Function description	Data out stage callbacks.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle • epnum: endpoint number
Return values	<ul style="list-style-type: none"> • None:

HAL_PCD_DataInStageCallback

Function name	void HAL_PCD_DataInStageCallback (PCD_HandleTypeDef * hpcd, uint8_t epnum)
Function description	Data IN stage callbacks.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle • epnum: endpoint number
Return values	<ul style="list-style-type: none"> • None:

HAL_PCD_SetupStageCallback

Function name	void HAL_PCD_SetupStageCallback (PCD_HandleTypeDef * hpcd)
Function description	Setup stage callback.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_PCD_SOFCallback

Function name	void HAL_PCD_SOFCallback (PCD_HandleTypeDef * hpcd)
Function description	USB Start Of Frame callbacks.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_PCD_ResetCallback

Function name	void HAL_PCD_ResetCallback (PCD_HandleTypeDef * hpcd)
Function description	USB Reset callbacks.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_PCD_SuspendCallback

Function name	void HAL_PCD_SuspendCallback (PCD_HandleTypeDef * hpcd)
Function description	Suspend event callbacks.

Parameters	<ul style="list-style-type: none">• hpcd: PCD handle
Return values	<ul style="list-style-type: none">• None:

HAL_PCD_ResumeCallback

Function name	void HAL_PCD_ResumeCallback (PCD_HandleTypeDef * hpcd)
Function description	Resume event callbacks.
Parameters	<ul style="list-style-type: none">• hpcd: PCD handle
Return values	<ul style="list-style-type: none">• None:

HAL_PCD_ISOOUTIncompleteCallback

Function name	void HAL_PCD_ISOOUTIncompleteCallback (PCD_HandleTypeDef * hpcd, uint8_t epnum)
Function description	Incomplete ISO OUT callbacks.
Parameters	<ul style="list-style-type: none">• hpcd: PCD handle• epnum: endpoint number
Return values	<ul style="list-style-type: none">• None:

HAL_PCD_ISOINIncompleteCallback

Function name	void HAL_PCD_ISOINIncompleteCallback (PCD_HandleTypeDef * hpcd, uint8_t epnum)
Function description	Incomplete ISO IN callbacks.
Parameters	<ul style="list-style-type: none">• hpcd: PCD handle• epnum: endpoint number
Return values	<ul style="list-style-type: none">• None:

HAL_PCD_ConnectCallback

Function name	void HAL_PCD_ConnectCallback (PCD_HandleTypeDef * hpcd)
Function description	Connection event callbacks.
Parameters	<ul style="list-style-type: none">• hpcd: PCD handle
Return values	<ul style="list-style-type: none">• None:

HAL_PCD_DisconnectCallback

Function name	void HAL_PCD_DisconnectCallback (PCD_HandleTypeDef * hpcd)
Function description	Disconnection event callbacks.
Parameters	<ul style="list-style-type: none">• hpcd: PCD handle
Return values	<ul style="list-style-type: none">• None:

HAL_PCD_DevConnect

Function name **HAL_StatusTypeDef HAL_PCD_DevConnect
(PCD_HandleTypeDef * hpcd)**

Function description Connect the USB device.

Parameters • **hpcd:** PCD handle

Return values • **HAL:** status

HAL_PCD_DevDisconnect

Function name **HAL_StatusTypeDef HAL_PCD_DevDisconnect
(PCD_HandleTypeDef * hpcd)**

Function description Disconnect the USB device.

Parameters • **hpcd:** PCD handle

Return values • **HAL:** status

HAL_PCD_SetAddress

Function name **HAL_StatusTypeDef HAL_PCD_SetAddress
(PCD_HandleTypeDef * hpcd, uint8_t address)**

Function description Set the USB Device address.

Parameters • **hpcd:** PCD handle

• **address:** new device address

Return values • **HAL:** status

HAL_PCD_EP_Open

Function name **HAL_StatusTypeDef HAL_PCD_EP_Open
(PCD_HandleTypeDef * hpcd, uint8_t ep_addr, uint16_t
ep_mps, uint8_t ep_type)**

Function description Open and configure an endpoint.

Parameters • **hpcd:** PCD handle

• **ep_addr:** endpoint address

• **ep_mps:** endpoint max packet size

• **ep_type:** endpoint type

Return values • **HAL:** status

HAL_PCD_EP_Close

Function name **HAL_StatusTypeDef HAL_PCD_EP_Close
(PCD_HandleTypeDef * hpcd, uint8_t ep_addr)**

Function description Deactivate an endpoint.

Parameters • **hpcd:** PCD handle

• **ep_addr:** endpoint address

Return values • **HAL:** status

HAL_PCD_EP_Receive

Function name	HAL_StatusTypeDef HAL_PCD_EP_Receive (PCD_HandleTypeDef * hpcd, uint8_t ep_addr, uint8_t * pBuf, uint32_t len)
Function description	Receive an amount of data.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle • ep_addr: endpoint address • pBuf: pointer to the reception buffer • len: amount of data to be received
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_PCD_EP_Transmit

Function name	HAL_StatusTypeDef HAL_PCD_EP_Transmit (PCD_HandleTypeDef * hpcd, uint8_t ep_addr, uint8_t * pBuf, uint32_t len)
Function description	Send an amount of data.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle • ep_addr: endpoint address • pBuf: pointer to the transmission buffer • len: amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_PCD_EP_GetRxCount

Function name	uint16_t HAL_PCD_EP_GetRxCount (PCD_HandleTypeDef * hpcd, uint8_t ep_addr)
Function description	Get Received Data Size.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle • ep_addr: endpoint address
Return values	<ul style="list-style-type: none"> • Data: Size

HAL_PCD_EP_SetStall

Function name	HAL_StatusTypeDef HAL_PCD_EP_SetStall (PCD_HandleTypeDef * hpcd, uint8_t ep_addr)
Function description	Set a STALL condition over an endpoint.
Parameters	<ul style="list-style-type: none"> • hpcd: PCD handle • ep_addr: endpoint address
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_PCD_EP_ClrStall

Function name	HAL_StatusTypeDef HAL_PCD_EP_ClrStall (PCD_HandleTypeDef * hpcd, uint8_t ep_addr)
Function description	Clear a STALL condition over in an endpoint.

Parameters	<ul style="list-style-type: none"> hpcd: PCD handle ep_addr: endpoint address
Return values	<ul style="list-style-type: none"> HAL: status

HAL_PCD_EP_Flush

Function name	HAL_StatusTypeDef HAL_PCD_EP_Flush (PCD_HandleTypeDef * hpcd, uint8_t ep_addr)
Function description	Flush an endpoint.
Parameters	<ul style="list-style-type: none"> hpcd: PCD handle ep_addr: endpoint address
Return values	<ul style="list-style-type: none"> HAL: status

HAL_PCD_ActivateRemoteWakeup

Function name	HAL_StatusTypeDef HAL_PCD_ActivateRemoteWakeup (PCD_HandleTypeDef * hpcd)
Function description	HAL_PCD_ActivateRemoteWakeup : active remote wakeup signalling.
Parameters	<ul style="list-style-type: none"> hpcd: PCD handle
Return values	<ul style="list-style-type: none"> HAL: status

HAL_PCD_DeActivateRemoteWakeup

Function name	HAL_StatusTypeDef HAL_PCD_DeActivateRemoteWakeup (PCD_HandleTypeDef * hpcd)
Function description	HAL_PCD_DeActivateRemoteWakeup : de-active remote wakeup signalling.
Parameters	<ul style="list-style-type: none"> hpcd: PCD handle
Return values	<ul style="list-style-type: none"> HAL: status

HAL_PCD_GetState

Function name	PCD_StateTypeDef HAL_PCD_GetState (PCD_HandleTypeDef * hpcd)
Function description	Return the PCD state.
Parameters	<ul style="list-style-type: none"> hpcd: : PCD handle
Return values	<ul style="list-style-type: none"> HAL: state

HAL_PCDEx_SetConnectionState

Function name	void HAL_PCDEx_SetConnectionState (PCD_HandleTypeDef * hpcd, uint8_t state)
Function description	Software Device Connection.
Parameters	<ul style="list-style-type: none"> hpcd: PCD handle

- **state:** Device state
- Return values • **None:**

30.3 PCD Firmware driver defines

30.3.1 PCD

PCD Core PHY

PCD_PHY_EMBEDDED

PCD Core Speed

PCD_SPEED_HIGH

PCD_SPEED_FULL

PCD ENDP

PCD_ENDP0

PCD_ENDP1

PCD_ENDP2

PCD_ENDP3

PCD_ENDP4

PCD_ENDP5

PCD_ENDP6

PCD_ENDP7

IS_PCD_ALL_INSTANCE

PCD Endpoint Kind

PCD_SNG_BUF

PCD_DBL_BUF

PCD EP0 MPS

DEP0CTL_MPS_64

DEP0CTL_MPS_32

DEP0CTL_MPS_16

DEP0CTL_MPS_8

PCD_EP0MPS_64

PCD_EP0MPS_32

PCD_EP0MPS_16

PCD_EP0MPS_08

PCD EP Type

PCD_EP_TYPE_CTRL

PCD_EP_TYPE_ISOC

PCD_EP_TYPE_BULK

PCD_EP_TYPE_INTR

PCD Exported Macros

`_HAL_PCD_GET_FLAG`

`_HAL_PCD_CLEAR_FLAG`

`_HAL_USB_WAKEUP_EXTI_ENABLE_IT`

`_HAL_USB_WAKEUP_EXTI_DISABLE_IT`

`_HAL_USB_WAKEUP_EXTI_GET_FLAG`

`_HAL_USB_WAKEUP_EXTI_CLEAR_FLAG`

`_HAL_USB_WAKEUP_EXTI_ENABLE_RISING_EDGE`

`_HAL_USB_WAKEUP_EXTI_ENABLE_FALLING_EDGE`

`_HAL_USB_WAKEUP_EXTI_ENABLE_RISING_FALLING_EDGE`

PCD_Exti_Line_Wakeup

`USB_WAKEUP_EXTI_LINE` External interrupt line 18 Connected to the USB FS EXTI Line

31 HAL PCD Extension Driver

31.1 PCDEEx Firmware driver API description

31.1.1 Peripheral Control functions

This section provides functions allowing to:

- Configure PMA for the EndPoint

This section contains the following APIs:

- [*HAL_PCDEEx_PMACConfig\(\)*](#)

31.1.2 Detailed description of functions

HAL_PCDEEx_PMACConfig

Function name	HAL_StatusTypeDef HAL_PCDEEx_PMACConfig (PCD_HandleTypeDef * hpcd, uint16_t ep_addr, uint16_t ep_kind, uint32_t pmaaddress)
Function description	Configure PMA for EP.
Parameters	<ul style="list-style-type: none">• hpcd: : Device instance• ep_addr: endpoint address• ep_kind: endpoint Kind USB_SNG_BUF: Single Buffer used USB_DBL_BUF: Double Buffer used• pmaaddress: EP address in The PMA: In case of single buffer endpoint this parameter is 16-bit value providing the address in PMA allocated to endpoint. In case of double buffer endpoint this parameter is a 32-bit value providing the endpoint buffer 0 address in the LSB part of 32-bit value and endpoint buffer 1 address in the MSB part of 32-bit value.
Return values	<ul style="list-style-type: none">• :: status

32 HAL PWR Generic Driver

32.1 PWR Firmware driver registers structures

32.1.1 PWR_PVDTTypeDef

Data Fields

- *uint32_t PVDLevel*
- *uint32_t Mode*

Field Documentation

- *uint32_t PWR_PVDTTypeDef::PVDLevel*

PVDLevel: Specifies the PVD detection level. This parameter can be a value of [**PWR_PVD_detection_level**](#)

- *uint32_t PWR_PVDTTypeDef::Mode*

Mode: Specifies the operating mode for the selected pins. This parameter can be a value of [**PWR_PVD_Mode**](#)

32.2 PWR Firmware driver API description

32.2.1 Initialization and de-initialization functions

After reset, the backup domain (RTC registers, RTC backup data registers) is protected against possible unwanted write accesses. To enable access to the RTC Domain and RTC registers, proceed as follows:

- Enable the Power Controller (PWR) APB1 interface clock using the `__HAL_RCC_PWR_CLK_ENABLE()` macro.
- Enable access to RTC domain using the `HAL_PWR_EnableBkUpAccess()` function.

This section contains the following APIs:

- [**HAL_PWR_DelInit\(\)**](#)
- [**HAL_PWR_EnableBkUpAccess\(\)**](#)
- [**HAL_PWR_DisableBkUpAccess\(\)**](#)

32.2.2 Peripheral Control functions

PVD configuration

- The PVD is used to monitor the VDD power supply by comparing it to a threshold selected by the PVD Level (PLS[2:0] bits in the PWR_CR).
- The PVD can use an external input analog voltage (PVD_IN) which is compared internally to VREFINT. The PVD_IN (PB7) has to be configured in Analog mode when PWR_PVDLevel_7 is selected (PLS[2:0] = 111).
- A PVDO flag is available to indicate if VDD/VDDA is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled. This is done through `__HAL_PWR_PVD_EXTI_ENABLE_IT()` macro.
- The PVD is stopped in Standby mode.

WakeUp pin configuration

- WakeUp pin is used to wake up the system from Standby mode. This pin is forced in input pull-down configuration and is active on rising edges.
- There are two or three WakeUp pins: WakeUp Pin 1 on PA.00. WakeUp Pin 2 on PC.13. WakeUp Pin 3 on PE.06. : Only on product with GPIOE available

Main and Backup Regulators configuration

Low Power modes configuration

The device features 5 low-power modes:

- Low power run mode: regulator in low power mode, limited clock frequency, limited number of peripherals running.
- Sleep mode: Cortex-M3 core stopped, peripherals kept running.
- Low power sleep mode: Cortex-M3 core stopped, limited clock frequency, limited number of peripherals running, regulator in low power mode.
- Stop mode: All clocks are stopped, regulator running, regulator in low power mode.
- Standby mode: VCORE domain powered off

Low power run mode

To further reduce the consumption when the system is in Run mode, the regulator can be configured in low power mode. In this mode, the system frequency should not exceed MSI frequency range1. In Low power run mode, all I/O pins keep the same state as in Run mode.

- Entry:
 - VCORE in range2
 - Decrease the system frequency tonot exceed the frequency of MSI frequency range1.
 - The regulator is forced in low power mode using the HAL_PWREx_EnableLowPowerRunMode() function.
- Exit:
 - The regulator is forced in Main regulator mode using the HAL_PWREx_DisableLowPowerRunMode() function.
 - Increase the system frequency if needed.

Sleep mode

- Entry: The Sleep mode is entered by using the HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFx) functions with
 - PWR_SLEEPENTRY_WFI: enter SLEEP mode with WFI instruction
 - PWR_SLEEPENTRY_WFE: enter SLEEP mode with WFE instruction
- Exit:
 - Any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode.

Low power sleep mode

- Entry: The Low power sleep mode is entered by using the HAL_PWR_EnterSLEEPMode(PWR_LOWPOWERREGULATOR_ON, PWR_SLEEPENTRY_WFx) functions with

- PWR_SLEEPENTRY_WFI: enter SLEEP mode with WFI instruction
- PWR_SLEEPENTRY_WFE: enter SLEEP mode with WFE instruction
- The Flash memory can be switched off by using the control bits (SLEEP_PD in the FLASH_ACR register). This reduces power consumption but increases the wake-up time.
- Exit:
 - If the WFI instruction was used to enter Low power sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Low power sleep mode. If the WFE instruction was used to enter Low power sleep mode, the MCU exits Sleep mode as soon as an event occurs.

Stop mode

The Stop mode is based on the Cortex-M3 deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode. In Stop mode, all clocks in the VCORE domain are stopped, the PLL, the MSI, the HSI and the HSE RC oscillators are disabled. Internal SRAM and register contents are preserved. To get the lowest consumption in Stop mode, the internal Flash memory also enters low power mode. When the Flash memory is in power-down mode, an additional startup delay is incurred when waking up from Stop mode. To minimize the consumption In Stop mode, VREFINT, the BOR, PVD, and temperature sensor can be switched off before entering Stop mode. They can be switched on again by software after exiting Stop mode using the ULP bit in the PWR_CR register. In Stop mode, all I/O pins keep the same state as in Run mode.

- Entry: The Stop mode is entered using the HAL_PWR_EnterSTOPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI) function with:
 - Main regulator ON.
 - Low Power regulator ON.
 - PWR_SLEEPENTRY_WFI: enter SLEEP mode with WFI instruction
 - PWR_SLEEPENTRY_WFE: enter SLEEP mode with WFE instruction
- Exit:
 - By issuing an interrupt or a wakeup event, the MSI RC oscillator is selected as system clock.

Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex-M3 deepsleep mode, with the voltage regulator disabled. The VCORE domain is consequently powered off. The PLL, the MSI, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for the RTC registers, RTC backup registers and Standby circuitry. To minimize the consumption In Standby mode, VREFINT, the BOR, PVD, and temperature sensor can be switched off before entering the Standby mode. They can be switched on again by software after exiting the Standby mode. function.

- Entry:
 - The Standby mode is entered using the HAL_PWR_EnterSTANDBYMode() function.
- Exit:
 - WKUP pin rising edge, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time-stamp event, external reset in NRST pin, IWDG reset.

Auto-wakeup (AWU) from low-power mode

The MCU can be woken up from low-power mode by an RTC Alarm event, an RTC Wakeup event, a tamper event, a time-stamp event, or a comparator event, without depending on an external interrupt (Auto-wakeup mode).

- RTC auto-wakeup (AWU) from the Stop mode
 - To wake up from the Stop mode with an RTC alarm event, it is necessary to:
 - Configure the EXTI Line 17 to be sensitive to rising edges (Interrupt or Event modes) and Enable the RTC Alarm Interrupt using the HAL_RTC_SetAlarm_IT() function
 - Configure the RTC to generate the RTC alarm using the HAL_RTC_Init() and HAL_RTC_SetTime() functions.
 - To wake up from the Stop mode with an RTC Tamper or time stamp event, it is necessary to:
 - Configure the EXTI Line 19 to be sensitive to rising edges (Interrupt or Event modes) and Enable the RTC Tamper or time stamp Interrupt using the HAL_RTCEx_SetTamper_IT() or HAL_RTCEx_SetTimeStamp_IT() functions.
 - To wake up from the Stop mode with an RTC WakeUp event, it is necessary to:
 - Configure the EXTI Line 20 to be sensitive to rising edges (Interrupt or Event modes) and Enable the RTC WakeUp Interrupt using the HAL_RTCEx_SetWakeUpTimer_IT() function.
 - Configure the RTC to generate the RTC WakeUp event using the HAL_RTCEx_SetWakeUpTimer() function.
- RTC auto-wakeup (AWU) from the Standby mode
 - To wake up from the Standby mode with an RTC alarm event, it is necessary to:
 - Enable the RTC Alarm Interrupt using the HAL_RTC_SetAlarm_IT() function.
 - Configure the RTC to generate the RTC alarm using the HAL_RTC_Init() and HAL_RTC_SetTime() functions.
 - To wake up from the Standby mode with an RTC Tamper or time stamp event, it is necessary to:
 - Enable the RTC Tamper or time stamp Interrupt and Configure the RTC to detect the tamper or time stamp event using the HAL_RTCEx_SetTimeStamp_IT() or HAL_RTCEx_SetTamper_IT()functions.
 - To wake up from the Standby mode with an RTC WakeUp event, it is necessary to:
 - Enable the RTC WakeUp Interrupt and Configure the RTC to generate the RTC WakeUp event using the HAL_RTCEx_SetWakeUpTimer_IT() and HAL_RTCEx_SetWakeUpTimer() functions.
- Comparator auto-wakeup (AWU) from the Stop mode
 - To wake up from the Stop mode with an comparator 1 or comparator 2 wakeup event, it is necessary to:
 - Configure the EXTI Line 21 or EXTI Line 22 for comparator to be sensitive to the selected edges (falling, rising or falling and rising) (Interrupt or Event modes) using the COMP functions.
 - Configure the comparator to generate the event.

This section contains the following APIs:

- [**HAL_PWR_ConfigPVD\(\)**](#)
- [**HAL_PWR_EnablePVD\(\)**](#)
- [**HAL_PWR_DisablePVD\(\)**](#)
- [**HAL_PWR_EnableWakeUpPin\(\)**](#)

- [*HAL_PWR_DisableWakeUpPin\(\)*](#)
- [*HAL_PWR_EnterSLEEPMode\(\)*](#)
- [*HAL_PWR_EnterSTOPMode\(\)*](#)
- [*HAL_PWR_EnterSTANDBYMode\(\)*](#)
- [*HAL_PWR_EnableSleepOnExit\(\)*](#)
- [*HAL_PWR_DisableSleepOnExit\(\)*](#)
- [*HAL_PWR_EnableSEVOnPend\(\)*](#)
- [*HAL_PWR_DisableSEVOnPend\(\)*](#)
- [*HAL_PWR_PVD_IRQHandler\(\)*](#)
- [*HAL_PWR_PVDCALLBACK\(\)*](#)

32.2.3 Detailed description of functions

HAL_PWR_DeInit

Function name	void HAL_PWR_DeInit (void)
Function description	Deinitializes the PWR peripheral registers to their default reset values.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Before calling this function, the VOS[1:0] bits should be configured to "10" and the system frequency has to be configured accordingly. To configure the VOS[1:0] bits, use the PWR_VoltageScalingConfig() function. • ULP and FWU bits are not reset by this function.

HAL_PWR_EnableBkUpAccess

Function name	void HAL_PWR_EnableBkUpAccess (void)
Function description	Enables access to the backup domain (RTC registers, RTC backup data registers).
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • If the HSE divided by 2, 4, 8 or 16 is used as the RTC clock, the Backup Domain Access should be kept enabled.

HAL_PWR_DisableBkUpAccess

Function name	void HAL_PWR_DisableBkUpAccess (void)
Function description	Disables access to the backup domain (RTC registers, RTC backup data registers).
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • If the HSE divided by 2, 4, 8 or 16 is used as the RTC clock, the Backup Domain Access should be kept enabled.

HAL_PWR_ConfigPVD

Function name	void HAL_PWR_ConfigPVD (PWR_PVDTTypeDef * sConfigPVD)
Function description	Configures the voltage threshold detected by the Power Voltage Detector(PVD).

Parameters	<ul style="list-style-type: none"> • sConfigPVD: pointer to an PWR_PVDTTypeDef structure that contains the configuration information for the PVD.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Refer to the electrical characteristics of your device datasheet for more details about the voltage threshold corresponding to each detection level.

HAL_PWR_EnablePVD

Function name	void HAL_PWR_EnablePVD (void)
Function description	Enables the Power Voltage Detector(PVD).
Return values	<ul style="list-style-type: none"> • None:

HAL_PWR_DisablePVD

Function name	void HAL_PWR_DisablePVD (void)
Function description	Disables the Power Voltage Detector(PVD).
Return values	<ul style="list-style-type: none"> • None:

HAL_PWR_EnableWakeUpPin

Function name	void HAL_PWR_EnableWakeUpPin (uint32_t WakeUpPinx)
Function description	Enables the WakeUp PINx functionality.
Parameters	<ul style="list-style-type: none"> • WakeUpPinx: Specifies the Power Wake-Up pin to enable. This parameter can be one of the following values: <ul style="list-style-type: none"> – PWR_WAKEUP_PIN1 – PWR_WAKEUP_PIN2 – PWR_WAKEUP_PIN3: Only on product with GPIOE available
Return values	<ul style="list-style-type: none"> • None:

HAL_PWR_DisableWakeUpPin

Function name	void HAL_PWR_DisableWakeUpPin (uint32_t WakeUpPinx)
Function description	Disables the WakeUp PINx functionality.
Parameters	<ul style="list-style-type: none"> • WakeUpPinx: Specifies the Power Wake-Up pin to disable. This parameter can be one of the following values: <ul style="list-style-type: none"> – PWR_WAKEUP_PIN1 – PWR_WAKEUP_PIN2 – PWR_WAKEUP_PIN3: Only on product with GPIOE available
Return values	<ul style="list-style-type: none"> • None:

HAL_PWR_EnterSTOPMode

Function name	void HAL_PWR_EnterSTOPMode (uint32_t Regulator, uint8_t STOPEntry)
---------------	---

Function description	Enters Stop mode.
Parameters	<ul style="list-style-type: none"> Regulator: Specifies the regulator state in Stop mode. This parameter can be one of the following values: <ul style="list-style-type: none"> PWR_MAINREGULATOR_ON: Stop mode with regulator ON PWR_LOWPOWERREGULATOR_ON: Stop mode with low power regulator ON STOPEntry: Specifies if Stop mode is entered with WFI or WFE instruction. This parameter can be one of the following values: <ul style="list-style-type: none"> PWR_STOPENTRY_WFI: Enter Stop mode with WFI instruction PWR_STOPENTRY_WFE: Enter Stop mode with WFE instruction
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> In Stop mode, all I/O pins keep the same state as in Run mode. When exiting Stop mode by using an interrupt or a wakeup event, MSI RC oscillator is selected as system clock. When the voltage regulator operates in low power mode, an additional startup delay is incurred when waking up from Stop mode. By keeping the internal regulator ON during Stop mode, the consumption is higher although the startup time is reduced.

HAL_PWR_EnterSLEEPMode

Function name	<code>void HAL_PWR_EnterSLEEPMode (uint32_t Regulator, uint8_t SLEEPEntry)</code>
Function description	Enters Sleep mode.
Parameters	<ul style="list-style-type: none"> Regulator: Specifies the regulator state in SLEEP mode. This parameter can be one of the following values: <ul style="list-style-type: none"> PWR_MAINREGULATOR_ON: SLEEP mode with regulator ON PWR_LOWPOWERREGULATOR_ON: SLEEP mode with low power regulator ON SLEEPEntry: Specifies if SLEEP mode is entered with WFI or WFE instruction. When WFI entry is used, tick interrupt have to be disabled if not desired as the interrupt wake up source. This parameter can be one of the following values: <ul style="list-style-type: none"> PWR_SLEEPENTRY_WFI: enter SLEEP mode with WFI instruction PWR_SLEEPENTRY_WFE: enter SLEEP mode with WFE instruction
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> In Sleep mode, all I/O pins keep the same state as in Run mode.

HAL_PWR_EnterSTANDBYMode

Function name	void HAL_PWR_EnterSTANDBYMode (void)
Function description	Enters Standby mode.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • In Standby mode, all I/O pins are high impedance except for: Reset pad (still available)RTC_AF1 pin (PC13) if configured for tamper, time-stamp, RTC Alarm out, or RTC clock calibration out.WKUP pin 1 (PA0) if enabled.WKUP pin 2 (PC13) if enabled.WKUP pin 3 (PE6) if enabled.

HAL_PWR_EnableSleepOnExit

Function name	void HAL_PWR_EnableSleepOnExit (void)
Function description	Indicates Sleep-On-Exit when returning from Handler mode to Thread mode.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Set SLEEPONEXIT bit of SCR register. When this bit is set, the processor re-enters SLEEP mode when an interruption handling is over. Setting this bit is useful when the processor is expected to run only on interruptions handling.

HAL_PWR_DisableSleepOnExit

Function name	void HAL_PWR_DisableSleepOnExit (void)
Function description	Disables Sleep-On-Exit feature when returning from Handler mode to Thread mode.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Clears SLEEPONEXIT bit of SCR register. When this bit is set, the processor re-enters SLEEP mode when an interruption handling is over.

HAL_PWR_EnableSEVOnPend

Function name	void HAL_PWR_EnableSEVOnPend (void)
Function description	Enables CORTEX M3 SEVONPEND bit.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Sets SEVONPEND bit of SCR register. When this bit is set, this causes WFE to wake up when an interrupt moves from inactive to pended.

HAL_PWR_DisableSEVOnPend

Function name	void HAL_PWR_DisableSEVOnPend (void)
Function description	Disables CORTEX M3 SEVONPEND bit.
Return values	<ul style="list-style-type: none"> • None:

- | | |
|-------|---|
| Notes | <ul style="list-style-type: none"> • Clears SEVONPEND bit of SCR register. When this bit is set, this causes WFE to wake up when an interrupt moves from inactive to pended. |
|-------|---|

HAL_PWR_PVD_IRQHandler

- | | |
|----------------------|---|
| Function name | void HAL_PWR_PVD_IRQHandler (void) |
| Function description | This function handles the PWR PVD interrupt request. |
| Return values | <ul style="list-style-type: none"> • None: |
| Notes | <ul style="list-style-type: none"> • This API should be called under the PVD_IRQHandler(). |

HAL_PWR_PVDCallback

- | | |
|----------------------|--|
| Function name | void HAL_PWR_PVDCallback (void) |
| Function description | PWR PVD interrupt callback. |
| Return values | <ul style="list-style-type: none"> • None: |

32.3 PWR Firmware driver defines

32.3.1 PWR

PWR CR Register alias address

LPSDSR_BIT_NUMBER

CR_LPSDSR_BB

DBP_BIT_NUMBER

CR_DBP_BB

LPRUN_BIT_NUMBER

CR_LPRUN_BB

PVDE_BIT_NUMBER

CR_PVDE_BB

FWU_BIT_NUMBER

CR_FWU_BB

ULP_BIT_NUMBER

CR_ULP_BB

PWR CSR Register alias address

CSR_EWUP_BB

PWR Exported Macros

_HAL_PWR_VOLTAGESCALING_CONFIG

Description:

- macros configure the main internal regulator output voltage.

Parameters:

- _REGULATOR__: specifies the



regulator output voltage to achieve a tradeoff between performance and power consumption when the device does not operate at the maximum frequency (refer to the datasheets for more details). This parameter can be one of the following values:

- PWR_REGULATOR_VOLTAGE_SCALE1: Regulator voltage output Scale 1 mode, System frequency up to 32 MHz.
- PWR_REGULATOR_VOLTAGE_SCALE2: Regulator voltage output Scale 2 mode, System frequency up to 16 MHz.
- PWR_REGULATOR_VOLTAGE_SCALE3: Regulator voltage output Scale 3 mode, System frequency up to 4.2 MHz

Return value:

- None

_HAL_PWR_GET_FLAG**Description:**

- Check PWR flag is set or not.

Parameters:

- _FLAG_: specifies the flag to check. This parameter can be one of the following values:
 - PWR_FLAG_WU: Wake Up flag. This flag indicates that a wakeup event was received from the WKUP pin or from the RTC alarm (Alarm B), RTC Tamper event, RTC TimeStamp event or RTC Wakeup. An additional wakeup event is detected if the WKUP pin is enabled (by setting the EWUP bit) when the WKUP pin level is already high.
 - PWR_FLAG_SB: StandBy flag. This flag indicates that the system was resumed from StandBy mode.
 - PWR_FLAG_PVDO: PVD Output. This flag is valid only if PVD is enabled by the HAL_PWR_EnablePVD() function. The PVD is stopped by Standby mode For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

- PWR_FLAG_VREFINTRDY: Internal voltage reference (VREFINT) ready flag. This bit indicates the state of the internal voltage reference, VREFINT.
- PWR_FLAG_VOS: Voltage Scaling select flag. A delay is required for the internal regulator to be ready after the voltage range is changed. The VOSF bit indicates that the regulator has reached the voltage level defined with bits VOS of PWR_CR register.
- PWR_FLAG_REGLP: Regulator LP flag. When the MCU exits from Low power run mode, this bit stays at 1 until the regulator is ready in main mode. A polling on this bit is recommended to wait for the regulator main mode. This bit is reset by hardware when the regulator is ready.

Return value:

- The: new state of __FLAG__ (TRUE or FALSE).

[__HAL_PWR_CLEAR_FLAG](#)**Description:**

- Clear the PWR's pending flags.

Parameters:

- __FLAG__: specifies the flag to clear. This parameter can be one of the following values:
 - PWR_FLAG_WU: Wake Up flag
 - PWR_FLAG_SB: StandBy flag

[__HAL_PWR_PVD_EXTI_ENABLE_IT](#)**Description:**

- Enable interrupt on PVD Exti Line 16.

Return value:

- None.

[__HAL_PWR_PVD_EXTI_DISABLE_IT](#)**Description:**

- Disable interrupt on PVD Exti Line 16.

Return value:

- None.

[__HAL_PWR_PVD_EXTI_ENABLE_EVENT](#)**Description:**

- Enable event on PVD Exti Line 16.

Return value:

- None.

`__HAL_PWR_PVD_EXTI_DISABLE_EVENT`

- Disable event on PVD Exti Line 16.

Return value:

- None.

`__HAL_PWR_PVD_EXTI_ENABLE_FALLING_EDGE`

Description:

- PVD EXTI line configuration: set falling edge trigger.

Return value:

- None.

`__HAL_PWR_PVD_EXTI_DISABLE_FALLING_EDGE`

Description:

- Disable the PVD Extended Interrupt Falling Trigger.

Return value:

- None.

`__HAL_PWR_PVD_EXTI_ENABLE_RISING_EDGE`

Description:

- PVD EXTI line configuration: set rising edge trigger.

Return value:

- None.

`__HAL_PWR_PVD_EXTI_DISABLE_RISING_EDGE`

Description:

- Disable the PVD Extended Interrupt Rising Trigger.

Return value:

- None.

`__HAL_PWR_PVD_EXTI_ENABLE_RISING_FALLING_EDGE`

Description:

- PVD EXTI line configuration: set rising & falling edge trigger.

Return value:

- None.

`__HAL_PWR_PVD_EXTI_DISABLE_RISING_FALLING_EDGE`

Description:

- Disable the PVD Extended Interrupt Rising & Falling Trigger.

Return value:

- None.

`__HAL_PWR_PVD_EXTI_GET_FLAG`

Description:

- Check whether the specified PVD EXTI interrupt flag is set or not.

Return value:

- EXTI: PVD Line Status.

`_HAL_PWR_PVD_EXTI_CLEAR_FLAG`**Description:**

- Clear the PVD EXTI flag.

Return value:

- None.

`_HAL_PWR_PVD_EXTI_GENERATE_SWIT`**Description:**

- Generate a Software interrupt on selected EXTI line.

Return value:

- None.

PWR Flag`PWR_FLAG_WU``PWR_FLAG_SB``PWR_FLAG_PVDO``PWR_FLAG_VREFINTRDY``PWR_FLAG_VOS``PWR_FLAG_REGLP`**PWR PVD detection level**`PWR_PVDLEVEL_0``PWR_PVDLEVEL_1``PWR_PVDLEVEL_2``PWR_PVDLEVEL_3``PWR_PVDLEVEL_4``PWR_PVDLEVEL_5``PWR_PVDLEVEL_6``PWR_PVDLEVEL_7`**PWR PVD Mode**`PWR_PVD_MODE_NORMAL`

basic mode is used

`PWR_PVD_MODE_IT_RISING`

External Interrupt Mode with Rising edge trigger detection

`PWR_PVD_MODE_IT_FALLING`

External Interrupt Mode with Falling edge trigger detection

`PWR_PVD_MODE_IT_RISING_FALLING`

External Interrupt Mode with Rising/Falling edge trigger detection

`PWR_PVD_MODE_EVENT_RISING`

Event Mode with Rising edge trigger

	detection
PWR_PVD_MODE_EVENT_FALLING	Event Mode with Falling edge trigger detection
PWR_PVD_MODE_EVENT_RISING_FALLING	Event Mode with Rising/Falling edge trigger detection
PWR Register alias address	
PWR_OFFSET	
PWR_CR_OFFSET	
PWR_CSR_OFFSET	
PWR_CR_OFFSET_BB	
PWR_CSR_OFFSET_BB	
PWR Regulator state in SLEEP/STOP mode	
PWR_MAINREGULATOR_ON	
PWR_LOWPOWERREGULATOR_ON	
PWR Regulator Voltage Scale	
PWR_REGULATOR_VOLTAGE_SCALE1	
PWR_REGULATOR_VOLTAGE_SCALE2	
PWR_REGULATOR_VOLTAGE_SCALE3	
PWR SLEEP mode entry	
PWR_SLEEPENTRY_WFI	
PWR_SLEEPENTRY_WFE	
PWR STOP mode entry	
PWR_STOPENTRY_WFI	
PWR_STOPENTRY_WFE	

33 HAL PWR Extension Driver

33.1 PWREx Firmware driver API description

33.1.1 Peripheral extended features functions

This section contains the following APIs:

- [**HAL_PWREx_GetVoltageRange\(\)**](#)
- [**HAL_PWREx_EnableFastWakeUp\(\)**](#)
- [**HAL_PWREx_DisableFastWakeUp\(\)**](#)
- [**HAL_PWREx_EnableUltraLowPower\(\)**](#)
- [**HAL_PWREx_DisableUltraLowPower\(\)**](#)
- [**HAL_PWREx_EnableLowPowerRunMode\(\)**](#)
- [**HAL_PWREx_DisableLowPowerRunMode\(\)**](#)

33.1.2 Detailed description of functions

HAL_PWREx_GetVoltageRange

Function name **uint32_t HAL_PWREx_GetVoltageRange (void)**

Function description Return Voltage Scaling Range.

Return values

- **VOS:** bit field (PWR_REGULATOR_VOLTAGE_SCALE1, PWR_REGULATOR_VOLTAGE_SCALE2 or PWR_REGULATOR_VOLTAGE_SCALE3)

HAL_PWREx_EnableFastWakeUp

Function name **void HAL_PWREx_EnableFastWakeUp (void)**

Function description Enables the Fast WakeUp from Ultra Low Power mode.

Return values

- **None:**

Notes

- This bit works in conjunction with ULP bit. Means, when ULP = 1 and FWU = 1 :VREFINT startup time is ignored when exiting from low power mode.

HAL_PWREx_DisableFastWakeUp

Function name **void HAL_PWREx_DisableFastWakeUp (void)**

Function description Disables the Fast WakeUp from Ultra Low Power mode.

Return values

- **None:**

HAL_PWREx_EnableUltraLowPower

Function name **void HAL_PWREx_EnableUltraLowPower (void)**

Function description Enables the Ultra Low Power mode.

Return values

- **None:**

HAL_PWREx_DisableUltraLowPower

Function name **void HAL_PWREx_DisableUltraLowPower (void)**

Function description Disables the Ultra Low Power mode.

Return values • **None:**

HAL_PWREx_EnableLowPowerRunMode

Function name **void HAL_PWREx_EnableLowPowerRunMode (void)**

Function description Enters the Low Power Run mode.

Return values • **None:**

Notes • Low power run mode can only be entered when VCORE is in range 2. In addition, the dynamic voltage scaling must not be used when Low power run mode is selected. Only Stop and Sleep modes with regulator configured in Low power mode is allowed when Low power run mode is selected.
• In Low power run mode, all I/O pins keep the same state as in Run mode.

HAL_PWREx_DisableLowPowerRunMode

Function name **HAL_StatusTypeDef
HAL_PWREx_DisableLowPowerRunMode (void)**

Function description Exits the Low Power Run mode.

Return values • **None:**

33.2 PWREx Firmware driver defines

33.2.1 PWREx

PWREx Wakeup Pins

PWR_WAKEUP_PIN1

PWR_WAKEUP_PIN2

PWR_WAKEUP_PIN3

IS_PWR_WAKEUP_PIN

34 HAL RCC Generic Driver

34.1 RCC Firmware driver registers structures

34.1.1 RCC_PLLInitTypeDef

Data Fields

- *uint32_t PLLState*
- *uint32_t PLLSource*
- *uint32_t PLLMUL*
- *uint32_t PLLDIV*

Field Documentation

- *uint32_t RCC_PLLInitTypeDef::PLLState*
PLLState: The new state of the PLL. This parameter can be a value of [RCC_PLL_Config](#)
- *uint32_t RCC_PLLInitTypeDef::PLLSource*
PLLSource: PLL entry clock source. This parameter must be a value of [RCC_PLL_Clock_Source](#)
- *uint32_t RCC_PLLInitTypeDef::PLLMUL*
PLLMUL: Multiplication factor for PLL VCO input clock This parameter must be a value of [RCC_PLL_Multiplication_Factor](#)
- *uint32_t RCC_PLLInitTypeDef::PLLDIV*
PLLDIV: Division factor for PLL VCO input clock This parameter must be a value of [RCC_PLL_Division_Factor](#)

34.1.2 RCC_OscInitTypeDef

Data Fields

- *uint32_t OscillatorType*
- *uint32_t HSEState*
- *uint32_t LSEState*
- *uint32_t HSISState*
- *uint32_t HSICalibrationValue*
- *uint32_t LSISState*
- *uint32_t MSISState*
- *uint32_t MSICalibrationValue*
- *uint32_t MSIClockRange*
- *RCC_PLLInitTypeDef PLL*

Field Documentation

- *uint32_t RCC_OscInitTypeDef::OscillatorType*
The oscillators to be configured. This parameter can be a value of [RCC_Oscillator_Type](#)
- *uint32_t RCC_OscInitTypeDef::HSEState*
The new state of the HSE. This parameter can be a value of [RCC_HSE_Config](#)
- *uint32_t RCC_OscInitTypeDef::LSEState*
The new state of the LSE. This parameter can be a value of [RCC_LSE_Config](#)
- *uint32_t RCC_OscInitTypeDef::HSISState*
The new state of the HSI. This parameter can be a value of [RCC_HSI_Config](#)

- ***uint32_t RCC_OsclInitTypeDef::HSICalibrationValue***
The HSI calibration trimming value (default is RCC_HSICALIBRATION_DEFAULT). This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1FU
- ***uint32_t RCC_OsclInitTypeDef::LSIState***
The new state of the LSI. This parameter can be a value of [RCC_LSI_Config](#)
- ***uint32_t RCC_OsclInitTypeDef::MSIState***
The new state of the MSI. This parameter can be a value of [RCC_MSI_Config](#)
- ***uint32_t RCC_OsclInitTypeDef::MSICalibrationValue***
The MSI calibration trimming value. (default is RCC_MSICALIBRATION_DEFAULT). This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFFU
- ***uint32_t RCC_OsclInitTypeDef::MSIClockRange***
The MSI frequency range. This parameter can be a value of [RCC_MSI_Clock_Range](#)
- ***RCC_PLLInitTypeDef RCC_OsclInitTypeDef::PLL***
PLL structure parameters

34.1.3 RCC_ClkInitTypeDef

Data Fields

- ***uint32_t ClockType***
- ***uint32_t SYSCLKSource***
- ***uint32_t AHBCLKDivider***
- ***uint32_t APB1CLKDivider***
- ***uint32_t APB2CLKDivider***

Field Documentation

- ***uint32_t RCC_ClkInitTypeDef::ClockType***
The clock to be configured. This parameter can be a value of [RCC_System_Clock_Type](#)
- ***uint32_t RCC_ClkInitTypeDef::SYSCLKSource***
The clock source (SYSCLKS) used as system clock. This parameter can be a value of [RCC_System_Clock_Source](#)
- ***uint32_t RCC_ClkInitTypeDef::AHBCLKDivider***
The AHB clock (HCLK) divider. This clock is derived from the system clock (SYSCLK). This parameter can be a value of [RCC_AHB_Clock_Source](#)
- ***uint32_t RCC_ClkInitTypeDef::APB1CLKDivider***
The APB1 clock (PCLK1) divider. This clock is derived from the AHB clock (HCLK). This parameter can be a value of [RCC_APB1_APB2_Clock_Source](#)
- ***uint32_t RCC_ClkInitTypeDef::APB2CLKDivider***
The APB2 clock (PCLK2) divider. This clock is derived from the AHB clock (HCLK). This parameter can be a value of [RCC_APB1_APB2_Clock_Source](#)

34.2 RCC Firmware driver API description

34.2.1 RCC specific features

After reset the device is running from multispeed internal oscillator clock (MSI 2.097MHz) with Flash 0 wait state and Flash prefetch buffer is disabled, and all peripherals are off except internal SRAM, Flash and JTAG.

- There is no prescaler on High speed (AHB) and Low speed (APB) buses; all peripherals mapped on these buses are running at MSI speed.
- The clock for all peripherals is switched off, except the SRAM and FLASH.
- All GPIOs are in input floating state, except the JTAG pins which are assigned to be used for debug purpose.

Once the device started from reset, the user application has to:

- Configure the clock source to be used to drive the System clock (if the application needs higher frequency/performance)
- Configure the System clock frequency and Flash settings
- Configure the AHB and APB buses prescalers
- Enable the clock for the peripheral(s) to be used
- Configure the clock source(s) for peripherals whose clocks are not derived from the System clock (I2S, RTC, ADC, USB OTG FS/SDIO/RNG) (*) SDIO only for STM32L1xxxD devices

34.2.2 RCC Limitations

A delay between an RCC peripheral clock enable and the effective peripheral enabling should be taken into account in order to manage the peripheral read/write from/to registers.

- This delay depends on the peripheral mapping.
 - AHB & APB peripherals, 1 dummy read is necessary

Workarounds:

1. For AHB & APB peripherals, a dummy read to the peripheral register has been inserted in each `__HAL_RCC_PPP_CLK_ENABLE()` macro.

34.2.3 Initialization and de-initialization functions

This section provides functions allowing to configure the internal/external oscillators (MSI, HSE, HSI, LSE, LSI, PLL, CSS and MCO) and the System buses clocks (SYSCLK, AHB, APB1 and APB2).

Internal/external clock and PLL configuration

1. MSI (Multispeed internal), Seven frequency ranges are available: 65.536 kHz, 131.072 kHz, 262.144 kHz, 524.288 kHz, 1.048 MHz, 2.097 MHz (default value) and 4.194 MHz.
2. HSI (high-speed internal), 16 MHz factory-trimmed RC used directly or through the PLL as System clock source.
3. LSI (low-speed internal), ~37 KHz low consumption RC used as IWDG and/or RTC clock source.
4. HSE (high-speed external), 1 to 24 MHz crystal oscillator used directly or through the PLL as System clock source. Can be used also as RTC clock source.
5. LSE (low-speed external), 32 KHz oscillator used as RTC clock source.
6. PLL (clocked by HSI or HSE), featuring different output clocks:
 - The first output is used to generate the high speed system clock (up to 32 MHz)
 - The second output is used to generate the clock for the USB OTG FS (48 MHz)
7. CSS (Clock security system), once enable using the macro `__HAL_RCC_CSS_ENABLE()` and if a HSE clock failure occurs(HSE used directly or through PLL as System clock source), the System clocks automatically switched to MSI and an interrupt is generated if enabled. The interrupt is linked to the Cortex-M3 NMI (Non-Maskable Interrupt) exception vector.
8. MCO1 (microcontroller clock output), used to output SYSCLK, HSI, LSI, MSI, LSE, HSE or PLL clock (through a configurable prescaler) on PA8 pin.

System, AHB and APB buses clocks configuration

1. Several clock sources can be used to drive the System clock (SYSCLK): MSI, HSI, HSE and PLL. The AHB clock (HCLK) is derived from System clock through configurable prescaler and used to clock the CPU, memory and peripherals mapped on AHB bus (DMA, GPIO...). APB1 (PCLK1) and APB2 (PCLK2) clocks are derived from AHB clock through configurable prescalers and used to clock the peripherals

- mapped on these buses. You can use "@ref HAL_RCC_GetSysClockFreq()" function to retrieve the frequencies of these clocks. All the peripheral clocks are derived from the System clock (SYSCLK) except: RTC: RTC clock can be derived either from the LSI, LSE or HSE clock divided by 2 to 16. You have to use @ref __HAL_RCC_RTC_CONFIG() and @ref __HAL_RCC_RTC_ENABLE() macros to configure this clock. LCD: LCD clock can be derived either from the LSI, LSE or HSE clock divided by 2 to 16. You have to use @ref __HAL_RCC_LCD_CONFIG() macros to configure this clock. USB OTG FS: USB OTG FS require a frequency equal to 48 MHz to work correctly. This clock is derived of the main PLL through PLL Multiplier. IWDG clock which is always the LSI clock.
2. The maximum frequency of the SYSCLK and HCLK is 32 MHz, PCLK2 32 MHz and PCLK1 32 MHz. Depending on the device voltage range, the maximum frequency should be adapted accordingly.

This section contains the following APIs:

- [`HAL_RCC_DelInit\(\)`](#)
- [`HAL_RCC_OscConfig\(\)`](#)
- [`HAL_RCC_ClockConfig\(\)`](#)

34.2.4 Peripheral Control functions

This subsection provides a set of functions allowing to control the RCC Clocks frequencies.

This section contains the following APIs:

- [`HAL_RCC_MCOConfig\(\)`](#)
- [`HAL_RCC_EnableCSS\(\)`](#)
- [`HAL_RCC_DisableCSS\(\)`](#)
- [`HAL_RCC_GetSysClockFreq\(\)`](#)
- [`HAL_RCC_GetHCLKFreq\(\)`](#)
- [`HAL_RCC_GetPCLK1Freq\(\)`](#)
- [`HAL_RCC_GetPCLK2Freq\(\)`](#)
- [`HAL_RCC_GetOscConfig\(\)`](#)
- [`HAL_RCC_GetClockConfig\(\)`](#)
- [`HAL_RCC_NMI_IRQHandler\(\)`](#)
- [`HAL_RCC_CSSCallback\(\)`](#)

34.2.5 Detailed description of functions

`HAL_RCC_DelInit`

Function name	<code>void HAL_RCC_DelInit (void)</code>
Function description	Resets the RCC clock configuration to the default reset state.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • The default reset state of the clock configuration is given below: MSI ON and used as system clock sourceHSI, HSE and PLL OFFAHB, APB1 and APB2 prescaler set to 1.CSS and MCO1 OFFAll interrupts disabled • This function does not modify the configuration of the Peripheral clocksLSI, LSE and RTC clocks

HAL_RCC_OscConfig

Function name	HAL_StatusTypeDef HAL_RCC_OscConfig((RCC_OscInitTypeDef * RCC_OscInitStruct)
Function description	Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef.
Parameters	<ul style="list-style-type: none"> • RCC_OscInitStruct: pointer to an RCC_OscInitTypeDef structure that contains the configuration information for the RCC Oscillators.
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • The PLL is not disabled when used as system clock. • Transitions LSE Bypass to LSE On and LSE On to LSE Bypass are not supported by this macro. User should request a transition to LSE Off first and then LSE On or LSE Bypass. • Transition HSE Bypass to HSE On and HSE On to HSE Bypass are not supported by this macro. User should request a transition to HSE Off first and then HSE On or HSE Bypass.

HAL_RCC_ClockConfig

Function name	HAL_StatusTypeDef HAL_RCC_ClockConfig((RCC_ClkInitTypeDef * RCC_ClkInitStruct, uint32_t FLatency)
Function description	Initializes the CPU, AHB and APB buses clocks according to the specified parameters in the RCC_ClkInitStruct.
Parameters	<ul style="list-style-type: none"> • RCC_ClkInitStruct: pointer to an RCC_OscInitTypeDef structure that contains the configuration information for the RCC peripheral. • FLatency: FLASH Latency The value of this parameter depend on device used within the same series
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • The SystemCoreClock CMSIS variable is used to store System Clock Frequency and updated by HAL_RCC_GetHCLKFreq() function called within this function • The MSI is used (enabled by hardware) as system clock source after start-up from Reset, wake-up from STOP and STANDBY mode, or in case of failure of the HSE used directly or indirectly as system clock (if the Clock Security System CSS is enabled). • A switch from one clock source to another occurs only if the target clock source is ready (clock stable after start-up delay or PLL locked). If a clock source which is not yet ready is selected, the switch will occur when the clock source will be ready. You can use HAL_RCC_GetClockConfig() function to know which clock is currently used as system clock source. • Depending on the device voltage range, the software has to set correctly HPRE[3:0] bits to ensure that HCLK not exceed the maximum allowed frequency (for more details refer to section above "Initialization/de-initialization functions")

HAL_RCC_MCOConfig

Function name	void HAL_RCC_MCOConfig (uint32_t RCC_MCOx, uint32_t RCC_MCOsource, uint32_t RCC_MCODiv)
Function description	Selects the clock source to output on MCO pin.
Parameters	<ul style="list-style-type: none"> • RCC_MCOx: specifies the output direction for the clock source. This parameter can be one of the following values: <ul style="list-style-type: none"> - RCC_MCO1 Clock source to output on MCO1 pin(PA8). • RCC_MCOsource: specifies the clock source to output. This parameter can be one of the following values: <ul style="list-style-type: none"> - RCC_MCO1SOURCE_NOCLOCK No clock selected as MCO clock - RCC_MCO1SOURCE_SYSCLK System clock selected as MCO clock - RCC_MCO1SOURCE_HSI HSI selected as MCO clock - RCC_MCO1SOURCE_HSE HSE selected as MCO clock - RCC_MCO1SOURCE_MSI MSI oscillator clock selected as MCO clock - RCC_MCO1SOURCE_PLLCLK PLL clock selected as MCO clock - RCC_MCO1SOURCE_LSI LSI clock selected as MCO clock - RCC_MCO1SOURCE_LSE LSE clock selected as MCO clock • RCC_MCODiv: specifies the MCO DIV. This parameter can be one of the following values: <ul style="list-style-type: none"> - RCC_MCODIV_1 no division applied to MCO clock - RCC_MCODIV_2 division by 2 applied to MCO clock - RCC_MCODIV_4 division by 4 applied to MCO clock - RCC_MCODIV_8 division by 8 applied to MCO clock - RCC_MCODIV_16 division by 16 applied to MCO clock
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • MCO pin should be configured in alternate function mode.

HAL_RCC_EnableCSS

Function name	void HAL_RCC_EnableCSS (void)
Function description	Enables the Clock Security System.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • If a failure is detected on the HSE oscillator clock, this oscillator is automatically disabled and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt, CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex-M3 NMI (Non-Maskable Interrupt) exception vector.

HAL_RCC_NMI_IRQHandler

Function name	void HAL_RCC_NMI_IRQHandler (void)
---------------	--

Function description	This function handles the RCC CSS interrupt request.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This API should be called under the NMI_Handler().

HAL_RCC_CSSCallback

Function name	void HAL_RCC_CSSCallback (void)
Function description	RCC Clock Security System interrupt callback.
Return values	<ul style="list-style-type: none"> • none:

HAL_RCC_DisableCSS

Function name	void HAL_RCC_DisableCSS (void)
Function description	Disables the Clock Security System.
Return values	<ul style="list-style-type: none"> • None:

HAL_RCC_GetSysClockFreq

Function name	uint32_t HAL_RCC_GetSysClockFreq (void)
Function description	Returns the SYSCLK frequency.
Return values	<ul style="list-style-type: none"> • SYSCLK: frequency
Notes	<ul style="list-style-type: none"> • The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source: • If SYSCLK source is MSI, function returns a value based on MSI Value as defined by the MSI range. • If SYSCLK source is HSI, function returns values based on HSI_VALUE(*) • If SYSCLK source is HSE, function returns a value based on HSE_VALUE(**) • If SYSCLK source is PLL, function returns a value based on HSE_VALUE(**) or HSI_VALUE(*) multiplied/divided by the PLL factors. • (*) HSI_VALUE is a constant defined in stm32l1xx_hal_conf.h file (default value 16 MHz) but the real value may vary depending on the variations in voltage and temperature. • (**) HSE_VALUE is a constant defined in stm32l1xx_hal_conf.h file (default value 8 MHz), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result. • The result of this function could be not correct when using fractional value for HSE crystal. • This function can be used by the user application to compute the baud-rate for the communication peripherals or configure other parameters. • Each time SYSCLK changes, this function must be called to update the right SYSCLK value. Otherwise, any configuration based on this function will be incorrect.

HAL_RCC_GetHCLKFreq

Function name	<code>uint32_t HAL_RCC_GetHCLKFreq (void)</code>
Function description	Returns the HCLK frequency.
Return values	<ul style="list-style-type: none"> • HCLK: frequency
Notes	<ul style="list-style-type: none"> • Each time HCLK changes, this function must be called to update the right HCLK value. Otherwise, any configuration based on this function will be incorrect. • The SystemCoreClock CMSIS variable is used to store System Clock Frequency and updated within this function

HAL_RCC_GetPCLK1Freq

Function name	<code>uint32_t HAL_RCC_GetPCLK1Freq (void)</code>
Function description	Returns the PCLK1 frequency.
Return values	<ul style="list-style-type: none"> • PCLK1: frequency
Notes	<ul style="list-style-type: none"> • Each time PCLK1 changes, this function must be called to update the right PCLK1 value. Otherwise, any configuration based on this function will be incorrect.

HAL_RCC_GetPCLK2Freq

Function name	<code>uint32_t HAL_RCC_GetPCLK2Freq (void)</code>
Function description	Returns the PCLK2 frequency.
Return values	<ul style="list-style-type: none"> • PCLK2: frequency
Notes	<ul style="list-style-type: none"> • Each time PCLK2 changes, this function must be called to update the right PCLK2 value. Otherwise, any configuration based on this function will be incorrect.

HAL_RCC_GetOscConfig

Function name	<code>void HAL_RCC_GetOscConfig (RCC_OscInitTypeDef * RCC_OscInitStruct)</code>
Function description	Configures the RCC_OscInitStruct according to the internal RCC configuration registers.
Parameters	<ul style="list-style-type: none"> • RCC_OscInitStruct: pointer to an RCC_OscInitTypeDef structure that will be configured.
Return values	<ul style="list-style-type: none"> • None:

HAL_RCC_GetClockConfig

Function name	<code>void HAL_RCC_GetClockConfig (RCC_ClkInitTypeDef * RCC_ClkInitStruct, uint32_t * pFLatency)</code>
Function description	Get the RCC_ClkInitStruct according to the internal RCC configuration registers.
Parameters	<ul style="list-style-type: none"> • RCC_ClkInitStruct: pointer to an RCC_ClkInitTypeDef structure that contains the current clock configuration.

- **pFLatency:** Pointer on the Flash Latency.
- Return values • **None:**

34.3 RCC Firmware driver defines

34.3.1 RCC

AHB Peripheral Clock Sleep Enable Disable Status

```
__HAL_RCC_GPIOA_IS_CLK_SLEEP_ENABLED  
__HAL_RCC_GPIOB_IS_CLK_SLEEP_ENABLED  
__HAL_RCC_GPIOC_IS_CLK_SLEEP_ENABLED  
__HAL_RCC_GPIOD_IS_CLK_SLEEP_ENABLED  
__HAL_RCC_GPIOH_IS_CLK_SLEEP_ENABLED  
__HAL_RCC_CRC_IS_CLK_SLEEP_ENABLED  
__HAL_RCC_FLITF_IS_CLK_SLEEP_ENABLED  
__HAL_RCC_DMA1_IS_CLK_SLEEP_ENABLED  
__HAL_RCC_GPIOA_IS_CLK_SLEEP_DISABLED  
__HAL_RCC_GPIOB_IS_CLK_SLEEP_DISABLED  
__HAL_RCC_GPIOC_IS_CLK_SLEEP_DISABLED  
__HAL_RCC_GPIOD_IS_CLK_SLEEP_DISABLED  
__HAL_RCC_GPIOH_IS_CLK_SLEEP_DISABLED  
__HAL_RCC_CRC_IS_CLK_SLEEP_DISABLED  
__HAL_RCC_FLITF_IS_CLK_SLEEP_DISABLED  
__HAL_RCC_DMA1_IS_CLK_SLEEP_DISABLED
```

AHB Clock Source

RCC_SYSCLK_DIV1	SYSCLK not divided
RCC_SYSCLK_DIV2	SYSCLK divided by 2
RCC_SYSCLK_DIV4	SYSCLK divided by 4
RCC_SYSCLK_DIV8	SYSCLK divided by 8
RCC_SYSCLK_DIV16	SYSCLK divided by 16
RCC_SYSCLK_DIV64	SYSCLK divided by 64
RCC_SYSCLK_DIV128	SYSCLK divided by 128
RCC_SYSCLK_DIV256	SYSCLK divided by 256
RCC_SYSCLK_DIV512	SYSCLK divided by 512

AHB Peripheral Clock Enable Disable Status

```
__HAL_RCC_GPIOA_IS_CLK_ENABLED  
__HAL_RCC_GPIOB_IS_CLK_ENABLED  
__HAL_RCC_GPIOC_IS_CLK_ENABLED
```

```
_HAL_RCC_GPIOD_IS_CLK_ENABLED  
_HAL_RCC_GPIOH_IS_CLK_ENABLED  
_HAL_RCC_CRC_IS_CLK_ENABLED  
_HAL_RCC_FLITF_IS_CLK_ENABLED  
_HAL_RCC_DMA1_IS_CLK_ENABLED  
_HAL_RCC_GPIOA_IS_CLK_DISABLED  
_HAL_RCC_GPIOB_IS_CLK_DISABLED  
_HAL_RCC_GPIOC_IS_CLK_DISABLED  
_HAL_RCC_GPIOD_IS_CLK_DISABLED  
_HAL_RCC_GPIOH_IS_CLK_DISABLED  
_HAL_RCC_CRC_IS_CLK_DISABLED  
_HAL_RCC_FLITF_IS_CLK_DISABLED  
_HAL_RCC_DMA1_IS_CLK_DISABLED
```

APB1 APB2 Clock Source

RCC_HCLK_DIV1	HCLK not divided
RCC_HCLK_DIV2	HCLK divided by 2
RCC_HCLK_DIV4	HCLK divided by 4
RCC_HCLK_DIV8	HCLK divided by 8
RCC_HCLK_DIV16	HCLK divided by 16

APB1 Clock Enable Disable

```
_HAL_RCC_TIM2_CLK_ENABLE  
_HAL_RCC_TIM3_CLK_ENABLE  
_HAL_RCC_TIM4_CLK_ENABLE  
_HAL_RCC_TIM6_CLK_ENABLE  
_HAL_RCC_TIM7_CLK_ENABLE  
_HAL_RCC_WWDG_CLK_ENABLE  
_HAL_RCC_SPI2_CLK_ENABLE  
_HAL_RCC_USART2_CLK_ENABLE  
_HAL_RCC_USART3_CLK_ENABLE  
_HAL_RCC_I2C1_CLK_ENABLE  
_HAL_RCC_I2C2_CLK_ENABLE  
_HAL_RCC_USB_CLK_ENABLE  
_HAL_RCC_PWR_CLK_ENABLE  
_HAL_RCC_DAC_CLK_ENABLE  
_HAL_RCC_COMP_CLK_ENABLE  
_HAL_RCC_TIM2_CLK_DISABLE
```

```
_HAL_RCC_TIM3_CLK_DISABLE  
_HAL_RCC_TIM4_CLK_DISABLE  
_HAL_RCC_TIM6_CLK_DISABLE  
_HAL_RCC_TIM7_CLK_DISABLE  
_HAL_RCC_WWDG_CLK_DISABLE  
_HAL_RCC_SPI2_CLK_DISABLE  
_HAL_RCC_USART2_CLK_DISABLE  
_HAL_RCC_USART3_CLK_DISABLE  
_HAL_RCC_I2C1_CLK_DISABLE  
_HAL_RCC_I2C2_CLK_DISABLE  
_HAL_RCC_USB_CLK_DISABLE  
_HAL_RCC_PWR_CLK_DISABLE  
_HAL_RCC_DAC_CLK_DISABLE  
_HAL_RCC_COMP_CLK_DISABLE
```

APB1 Peripheral Clock Sleep Enable Disable Status

```
_HAL_RCC_TIM2_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_TIM3_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_TIM4_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_TIM6_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_TIM7_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_WWDG_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_SPI2_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_USART2_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_USART3_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_I2C1_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_I2C2_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_USB_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_PWR_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_DAC_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_COMP_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_TIM2_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_TIM3_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_TIM4_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_TIM6_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_TIM7_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_WWDG_IS_CLK_SLEEP_DISABLED
```

```
_HAL_RCC_SPI2_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_USART2_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_USART3_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_I2C1_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_I2C2_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_USB_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_PWR_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_DAC_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_COMP_IS_CLK_SLEEP_DISABLED
```

APB1 Force Release Reset

```
_HAL_RCC_APB1_FORCE_RESET  
_HAL_RCC_TIM2_FORCE_RESET  
_HAL_RCC_TIM3_FORCE_RESET  
_HAL_RCC_TIM4_FORCE_RESET  
_HAL_RCC_TIM6_FORCE_RESET  
_HAL_RCC_TIM7_FORCE_RESET  
_HAL_RCC_WWDG_FORCE_RESET  
_HAL_RCC_SPI2_FORCE_RESET  
_HAL_RCC_USART2_FORCE_RESET  
_HAL_RCC_USART3_FORCE_RESET  
_HAL_RCC_I2C1_FORCE_RESET  
_HAL_RCC_I2C2_FORCE_RESET  
_HAL_RCC_USB_FORCE_RESET  
_HAL_RCC_PWR_FORCE_RESET  
_HAL_RCC_DAC_FORCE_RESET  
_HAL_RCC_COMP_FORCE_RESET  
_HAL_RCC_APB1_RELEASE_RESET  
_HAL_RCC_TIM2_RELEASE_RESET  
_HAL_RCC_TIM3_RELEASE_RESET  
_HAL_RCC_TIM4_RELEASE_RESET  
_HAL_RCC_TIM6_RELEASE_RESET  
_HAL_RCC_TIM7_RELEASE_RESET  
_HAL_RCC_WWDG_RELEASE_RESET  
_HAL_RCC_SPI2_RELEASE_RESET  
_HAL_RCC_USART2_RELEASE_RESET  
_HAL_RCC_USART3_RELEASE_RESET
```

```
_HAL_RCC_I2C1_RELEASE_RESET  
_HAL_RCC_I2C2_RELEASE_RESET  
_HAL_RCC_USB_RELEASE_RESET  
_HAL_RCC_PWR_RELEASE_RESET  
_HAL_RCC_DAC_RELEASE_RESET  
_HAL_RCC_COMP_RELEASE_RESET  
APB1 Peripheral Clock Enable Disable Status  
_HAL_RCC_TIM2_IS_CLK_ENABLED  
_HAL_RCC_TIM3_IS_CLK_ENABLED  
_HAL_RCC_TIM4_IS_CLK_ENABLED  
_HAL_RCC_TIM6_IS_CLK_ENABLED  
_HAL_RCC_TIM7_IS_CLK_ENABLED  
_HAL_RCC_WWDG_IS_CLK_ENABLED  
_HAL_RCC_SPI2_IS_CLK_ENABLED  
_HAL_RCC_USART2_IS_CLK_ENABLED  
_HAL_RCC_USART3_IS_CLK_ENABLED  
_HAL_RCC_I2C1_IS_CLK_ENABLED  
_HAL_RCC_I2C2_IS_CLK_ENABLED  
_HAL_RCC_USB_IS_CLK_ENABLED  
_HAL_RCC_PWR_IS_CLK_ENABLED  
_HAL_RCC_DAC_IS_CLK_ENABLED  
_HAL_RCC_COMP_IS_CLK_ENABLED  
_HAL_RCC_TIM2_IS_CLK_DISABLED  
_HAL_RCC_TIM3_IS_CLK_DISABLED  
_HAL_RCC_TIM4_IS_CLK_DISABLED  
_HAL_RCC_TIM6_IS_CLK_DISABLED  
_HAL_RCC_TIM7_IS_CLK_DISABLED  
_HAL_RCC_WWDG_IS_CLK_DISABLED  
_HAL_RCC_SPI2_IS_CLK_DISABLED  
_HAL_RCC_USART2_IS_CLK_DISABLED  
_HAL_RCC_USART3_IS_CLK_DISABLED  
_HAL_RCC_I2C1_IS_CLK_DISABLED  
_HAL_RCC_I2C2_IS_CLK_DISABLED  
_HAL_RCC_USB_IS_CLK_DISABLED  
_HAL_RCC_PWR_IS_CLK_DISABLED  
_HAL_RCC_DAC_IS_CLK_DISABLED
```

`_HAL_RCC_COMP_IS_CLK_DISABLED`

APB2 Clock Enable Disable

`_HAL_RCC_SYSCFG_CLK_ENABLE`

`_HAL_RCC_TIM9_CLK_ENABLE`

`_HAL_RCC_TIM10_CLK_ENABLE`

`_HAL_RCC_TIM11_CLK_ENABLE`

`_HAL_RCC_ADC1_CLK_ENABLE`

`_HAL_RCC_SPI1_CLK_ENABLE`

`_HAL_RCC_USART1_CLK_ENABLE`

`_HAL_RCC_SYSCFG_CLK_DISABLE`

`_HAL_RCC_TIM9_CLK_DISABLE`

`_HAL_RCC_TIM10_CLK_DISABLE`

`_HAL_RCC_TIM11_CLK_DISABLE`

`_HAL_RCC_ADC1_CLK_DISABLE`

`_HAL_RCC_SPI1_CLK_DISABLE`

`_HAL_RCC_USART1_CLK_DISABLE`

APB2 Peripheral Clock Sleep Enable Disable Status

`_HAL_RCC_SYSCFG_IS_CLK_SLEEP_ENABLED`

`_HAL_RCC_TIM9_IS_CLK_SLEEP_ENABLED`

`_HAL_RCC_TIM10_IS_CLK_SLEEP_ENABLED`

`_HAL_RCC_TIM11_IS_CLK_SLEEP_ENABLED`

`_HAL_RCC_ADC1_IS_CLK_SLEEP_ENABLED`

`_HAL_RCC_SPI1_IS_CLK_SLEEP_ENABLED`

`_HAL_RCC_USART1_IS_CLK_SLEEP_ENABLED`

`_HAL_RCC_SYSCFG_IS_CLK_SLEEP_DISABLED`

`_HAL_RCC_TIM9_IS_CLK_SLEEP_DISABLED`

`_HAL_RCC_TIM10_IS_CLK_SLEEP_DISABLED`

`_HAL_RCC_TIM11_IS_CLK_SLEEP_DISABLED`

`_HAL_RCC_ADC1_IS_CLK_SLEEP_DISABLED`

`_HAL_RCC_SPI1_IS_CLK_SLEEP_DISABLED`

`_HAL_RCC_USART1_IS_CLK_SLEEP_DISABLED`

APB2 Force Release Reset

`_HAL_RCC_APB2_FORCE_RESET`

`_HAL_RCC_SYSCFG_FORCE_RESET`

`_HAL_RCC_TIM9_FORCE_RESET`

`_HAL_RCC_TIM10_FORCE_RESET`

_HAL_RCC_TIM11_FORCE_RESET
_HAL_RCC_ADC1_FORCE_RESET
_HAL_RCC_SPI1_FORCE_RESET
_HAL_RCC_USART1_FORCE_RESET
_HAL_RCC_APB2_RELEASE_RESET
_HAL_RCC_SYSCFG_RELEASE_RESET
_HAL_RCC_TIM9_RELEASE_RESET
_HAL_RCC_TIM10_RELEASE_RESET
_HAL_RCC_TIM11_RELEASE_RESET
_HAL_RCC_ADC1_RELEASE_RESET
_HAL_RCC_SPI1_RELEASE_RESET
_HAL_RCC_USART1_RELEASE_RESET

APB2 Peripheral Clock Enable Disable Status

_HAL_RCC_SYSCFG_IS_CLK_ENABLED
_HAL_RCC_TIM9_IS_CLK_ENABLED
_HAL_RCC_TIM10_IS_CLK_ENABLED
_HAL_RCC_TIM11_IS_CLK_ENABLED
_HAL_RCC_ADC1_IS_CLK_ENABLED
_HAL_RCC_SPI1_IS_CLK_ENABLED
_HAL_RCC_USART1_IS_CLK_ENABLED
_HAL_RCC_SYSCFG_IS_CLK_DISABLED
_HAL_RCC_TIM9_IS_CLK_DISABLED
_HAL_RCC_TIM10_IS_CLK_DISABLED
_HAL_RCC_TIM11_IS_CLK_DISABLED
_HAL_RCC_ADC1_IS_CLK_DISABLED
_HAL_RCC_SPI1_IS_CLK_DISABLED
_HAL_RCC_USART1_IS_CLK_DISABLED

BitAddress AliasRegion

RCC_CR_OFFSET_BB
RCC_CFGR_OFFSET_BB
RCC_CIR_OFFSET_BB
RCC_CSR_OFFSET_BB
RCC_HSION_BIT_NUMBER
RCC_CR_HSION_BB
RCC_MSION_BIT_NUMBER
RCC_CR_MSION_BB

RCC_HSEON_BIT_NUMBER
RCC_CR_HSEON_BB
RCC_CSSON_BIT_NUMBER
RCC_CR_CSSON_BB
RCC_PLLON_BIT_NUMBER
RCC_CR_PLLON_BB
RCC_LSION_BIT_NUMBER
RCC_CSR_LSION_BB
RCC_RMVF_BIT_NUMBER
RCC_CSR_RMVF_BB
RCC_LSEON_BIT_NUMBER
RCC_CSR_LSEON_BB
RCC_LSEBYP_BIT_NUMBER
RCC_CSR_LSEBYP_BB
RCC_RTCEN_BIT_NUMBER
RCC_CSR_RTCEN_BB
RCC_RTCRST_BIT_NUMBER
RCC_CSR_RTCRST_BB

Flags

RCC_FLAG_HSIRDY	Internal High Speed clock ready flag
RCC_FLAG_MSIRDY	MSI clock ready flag
RCC_FLAG_HSERDY	External High Speed clock ready flag
RCC_FLAG_PLLRDY	PLL clock ready flag
RCC_FLAG_LSIRDY	Internal Low Speed oscillator Ready
RCC_FLAG_LSECSS	CSS on LSE failure Detection
RCC_FLAG_OBLRST	Options bytes loading reset flag
RCC_FLAG_PINRST	PIN reset flag
RCC_FLAG_PORRST	POR/PDR reset flag
RCC_FLAG_SFTRST	Software Reset flag
RCC_FLAG_IWDGRST	Independent Watchdog reset flag
RCC_FLAG_WWDGRST	Window watchdog reset flag
RCC_FLAG_LPWRRST	Low-Power reset flag
RCC_FLAG_LSERDY	External Low Speed oscillator Ready

Flags Interrupts Management

__HAL_RCC_ENABLE_IT	Description:
	<ul style="list-style-type: none">Enable RCC interrupt.

Parameters:

- __INTERRUPT__: specifies the RCC interrupt sources to be enabled. This parameter can be any combination of the following values:
 - RCC_IT_LSIRDY LSI ready interrupt
 - RCC_IT_LSERDY LSE ready interrupt
 - RCC_IT_HSIRDY HSI ready interrupt
 - RCC_IT_HSERDY HSE ready interrupt
 - RCC_IT_PLLRDY main PLL ready interrupt
 - RCC_IT_MSIRDY MSI ready interrupt
 - RCC_IT_LSECSS LSE CSS interrupt (not available for STM32L100xB || STM32L151xB || STM32L152xB devices)

[__HAL_RCC_DISABLE_IT](#)**Description:**

- Disable RCC interrupt.

Parameters:

- __INTERRUPT__: specifies the RCC interrupt sources to be disabled. This parameter can be any combination of the following values:
 - RCC_IT_LSIRDY LSI ready interrupt
 - RCC_IT_LSERDY LSE ready interrupt
 - RCC_IT_HSIRDY HSI ready interrupt
 - RCC_IT_HSERDY HSE ready interrupt
 - RCC_IT_PLLRDY main PLL ready interrupt
 - RCC_IT_MSIRDY MSI ready interrupt
 - RCC_IT_LSECSS LSE CSS interrupt (not available for STM32L100xB || STM32L151xB || STM32L152xB devices)

[__HAL_RCC_CLEAR_IT](#)**Description:**

- Clear the RCC's interrupt pending bits.

Parameters:

- __INTERRUPT__: specifies the interrupt pending bit to clear. This parameter can be any combination of the following values:
 - RCC_IT_LSIRDY LSI ready interrupt.
 - RCC_IT_LSERDY LSE ready interrupt.
 - RCC_IT_HSIRDY HSI ready interrupt.
 - RCC_IT_HSERDY HSE ready interrupt.
 - RCC_IT_PLLRDY Main PLL ready interrupt.
 - RCC_IT_MSIRDY MSI ready interrupt
 - RCC_IT_LSECSS LSE CSS interrupt

(not available for STM32L100xB || STM32L151xB || STM32L152xB devices)

- RCC_IT_CSS Clock Security System interrupt

__HAL_RCC_GET_IT**Description:**

- Check the RCC's interrupt has occurred or not.

Parameters:

- __INTERRUPT__: specifies the RCC interrupt source to check. This parameter can be one of the following values:
 - RCC_IT_LSIRDY LSI ready interrupt.
 - RCC_IT_LSERDY LSE ready interrupt.
 - RCC_IT_HSIRDY HSI ready interrupt.
 - RCC_IT_HSERDY HSE ready interrupt.
 - RCC_IT_PLLRDY Main PLL ready interrupt.
 - RCC_IT_MSIRDY MSI ready interrupt
 - RCC_IT_LSECSS LSE CSS interrupt (not available for STM32L100xB || STM32L151xB || STM32L152xB devices)
 - RCC_IT_CSS Clock Security System interrupt

Return value:

- The new state of __INTERRUPT__ (TRUE or FALSE).

__HAL_RCC_CLEAR_RESET_FLAGS

The reset flags are RCC_FLAG_PINRST, RCC_FLAG_PORRST, RCC_FLAG_SFTRST, RCC_FLAG_IWDGRST, RCC_FLAG_WWDGRST, RCC_FLAG_LPWRRST

__HAL_RCC_GET_FLAG**Description:**

- Check RCC flag is set or not.

Parameters:

- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
 - RCC_FLAG_HSIRDY HSI oscillator clock ready.
 - RCC_FLAG_MSIRDY MSI oscillator clock ready.
 - RCC_FLAG_HSERDY HSE oscillator clock ready.
 - RCC_FLAG_PLLRDY Main PLL clock ready.
 - RCC_FLAG_LSERDY LSE oscillator

- clock ready.
- RCC_FLAG_LSECSS CSS on LSE failure Detection (*)
- RCC_FLAG_LSIRDY LSI oscillator clock ready.
- RCC_FLAG_OBLRST Option Byte Load reset
- RCC_FLAG_PINRST Pin reset.
- RCC_FLAG_PORRST POR/PDR reset.
- RCC_FLAG_SFTRST Software reset.
- RCC_FLAG_IWDGRST Independent Watchdog reset.
- RCC_FLAG_WWDGRST Window Watchdog reset.
- RCC_FLAG_LPWRRST Low Power reset.

Return value:

- The: new state of __FLAG__ (TRUE or FALSE).

Notes:

- (*) This bit is available in high and medium+ density devices only.

Get Clock source[__HAL_RCC_SYSCLK_CONFIG](#)**Description:**

- Macro to configure the system clock source.

Parameters:

- __SYSCLKSOURCE__: specifies the system clock source. This parameter can be one of the following values:
 - RCC_SYSCLKSOURCE_MSI MSI oscillator is used as system clock source.
 - RCC_SYSCLKSOURCE_HSI HSI oscillator is used as system clock source.
 - RCC_SYSCLKSOURCE_HSE HSE oscillator is used as system clock source.
 - RCC_SYSCLKSOURCE_PLLCLK PLL output is used as system clock source.

[__HAL_RCC_GET_SYSCLK_SOURCE](#)**Description:**

- Macro to get the clock source used as system clock.

Return value:

- The: clock source used as system clock. The returned value can be one of the following:
 - RCC_SYSCLKSOURCE_STATUS_MSI MSI used as system clock
 - RCC_SYSCLKSOURCE_STATUS_HSI HSI used as system clock



- RCC_SYSCLKSOURCE_STATUS_HSE
HSE used as system clock
- RCC_SYSCLKSOURCE_STATUS_PLLCL
K PLL used as system clock

RTC HSE Prescaler

<code>RCC_RTC_HSE_DIV_2</code>	HSE is divided by 2 for RTC clock
<code>RCC_RTC_HSE_DIV_4</code>	HSE is divided by 4 for RTC clock
<code>RCC_RTC_HSE_DIV_8</code>	HSE is divided by 8 for RTC clock
<code>RCC_RTC_HSE_DIV_16</code>	HSE is divided by 16 for RTC clock

HSE Config

<code>RCC_HSE_OFF</code>	HSE clock deactivation
<code>RCC_HSE_ON</code>	HSE clock activation
<code>RCC_HSE_BYPASS</code>	External clock source for HSE clock

HSE Configuration

_HAL_RCC_HSE_CONFIG Description:

- Macro to configure the External High Speed oscillator (HSE).

Parameters:

- _STATE_: specifies the new state of the HSE. This parameter can be one of the following values:
 - `RCC_HSE_OFF` turn OFF the HSE oscillator, HSERDY flag goes low after 6 HSE oscillator clock cycles.
 - `RCC_HSE_ON` turn ON the HSE oscillator
 - `RCC_HSE_BYPASS` HSE oscillator bypassed with external clock

Notes:

- Transition HSE Bypass to HSE On and HSE On to HSE Bypass are not supported by this macro. User should request a transition to HSE Off first and then HSE On or HSE Bypass. After enabling the HSE (`RCC_HSE_ON` or `RCC_HSE_Bypass`), the application software should wait on HSERDY flag to be set indicating that HSE clock is stable and can be used to clock the PLL and/or system clock. HSE state can not be changed if it is used directly or through the PLL as system clock. In this case, you have to select another source of the system clock then change the HSE state (ex. disable it). The HSE is stopped by hardware when entering STOP and STANDBY modes. This function reset the CSSON bit, so if the clock security system(CSS) was previously enabled you have to enable it again after calling this function.

HSI Config

<code>RCC_HSI_OFF</code>	HSI clock deactivation
--------------------------	------------------------

`RCC_HSI_ON` HSI clock activation

`RCC_HSICALIBRATION_DEFAULT`

HSI Configuration

`_HAL_RCC_HSI_ENABLE`

Notes:

- The HSI is stopped by hardware when entering STOP and STANDBY modes. HSI can not be stopped if it is used as system clock source. In this case, you have to select another source of the system clock then stop the HSI. After enabling the HSI, the application software should wait on HSIRDY flag to be set indicating that HSI clock is stable and can be used as system clock source. When the HSI is stopped, HSIRDY flag goes low after 6 HSI oscillator clock cycles.

`_HAL_RCC_HSI_DISABLE`

`_HAL_RCC_HSI_CALIBRATIONVALUE_ADJUST`

Description:

- Macro to adjust the Internal High Speed oscillator (HSI) calibration value.

Parameters:

- `_HSICALIBRATIONVALUE_`: specifies the calibration trimming value. (default is `RCC_HSICALIBRATION_DEFAULT`). This parameter must be a number between 0 and 0x1F.

Notes:

- The calibration is used to compensate for the variations in voltage and temperature that influence the frequency of the internal HSI RC.

Interrupts

`RCC_IT_LSIRDY` LSI Ready Interrupt flag

`RCC_IT_LSERDY` LSE Ready Interrupt flag

`RCC_IT_HSIRDY` HSI Ready Interrupt flag

`RCC_IT_HSERDY` HSE Ready Interrupt flag

`RCC_IT_PLLRDY` PLL Ready Interrupt flag



<code>RCC_IT_MSIRDY</code>	MSI Ready Interrupt flag
<code>RCC_IT_LSECSS</code>	LSE Clock Security System Interrupt flag
<code>RCC_IT_CSS</code>	Clock Security System Interrupt flag
<i>LSE Config</i>	
<code>RCC_LSE_OFF</code>	LSE clock deactivation
<code>RCC_LSE_ON</code>	LSE clock activation
<code>RCC_LSE_BYPASS</code>	External clock source for LSE clock

LSE Configuration

`__HAL_RCC_LSE_CONFIG` **Description:**

- Macro to configure the External Low Speed oscillator (LSE).

Parameters:

- `__STATE__`: specifies the new state of the LSE. This parameter can be one of the following values:
 - `RCC_LSE_OFF` turn OFF the LSE oscillator, LSERDY flag goes low after 6 LSE oscillator clock cycles.
 - `RCC_LSE_ON` turn ON the LSE oscillator.
 - `RCC_LSE_BYPASS` LSE oscillator bypassed with external clock.

Notes:

- Transitions LSE Bypass to LSE On and LSE On to LSE Bypass are not supported by this macro. As the LSE is in the Backup domain and write access is denied to this domain after reset, you have to enable write access using `HAL_PWR_EnableBkUpAccess()` function before to configure the LSE (to be done once after reset). After enabling the LSE (`RCC_LSE_ON` or `RCC_LSE_BYPASS`), the application software should wait on LSERDY flag to be set indicating that LSE clock is stable and can be used to clock the RTC.

LSI Config

<code>RCC_LSI_OFF</code>	LSI clock deactivation
<code>RCC_LSI_ON</code>	LSI clock activation

LSI Configuration

`__HAL_RCC_LSI_ENABLE` **Notes:**

- After enabling the LSI, the application software should wait on LSIRDY flag to be set indicating that LSI clock is stable and can be used to clock the IWDG and/or the RTC.

`__HAL_RCC_LSI_DISABLE` **Notes:**

- LSI can not be disabled if the IWDG is running. When the LSI is stopped, LSIRDY flag goes low after 6 LSI

oscillator clock cycles.

MCO1 Clock Source

RCC_MCO1SOURCE_NOCLOCK
RCC_MCO1SOURCE_SYSCLK
RCC_MCO1SOURCE_MSI
RCC_MCO1SOURCE_HSI
RCC_MCO1SOURCE_LSE
RCC_MCO1SOURCE_LSI
RCC_MCO1SOURCE_HSE
RCC_MCO1SOURCE_PLLCLK

MCO Clock Prescaler

RCC_MCODIV_1
RCC_MCODIV_2
RCC_MCODIV_4
RCC_MCODIV_8
RCC_MCODIV_16

MCO Index

RCC_MCO1

RCC_MCO MCO1 to be compliant with other families with 2 MCOs

MSI Clock Range

RCC_MSIRANGE_0	MSI = 65.536 KHz
RCC_MSIRANGE_1	MSI = 131.072 KHz
RCC_MSIRANGE_2	MSI = 262.144 KHz
RCC_MSIRANGE_3	MSI = 524.288 KHz
RCC_MSIRANGE_4	MSI = 1.048 MHz
RCC_MSIRANGE_5	MSI = 2.097 MHz
RCC_MSIRANGE_6	MSI = 4.194 MHz

MSI Config

RCC_MSI_OFF
RCC_MSI_ON
RCC_MSICALIBRATION_DEFAULT

MSI Configuration

__HAL_RCC_MSI_ENABLE

Notes:

- After enabling the MSI, the application software should wait on MSIRDY flag to be set indicating that MSI clock is

stable and can be used as system clock source.

__HAL_RCC_MSI_DISABLE

Notes:

- The MSI is stopped by hardware when entering STOP and STANDBY modes. It is used (enabled by hardware) as system clock source after startup from Reset, wakeup from STOP and STANDBY mode, or in case of failure of the HSE used directly or indirectly as system clock (if the Clock Security System CSS is enabled). MSI can not be stopped if it is used as system clock source. In this case, you have to select another source of the system clock then stop the MSI. When the MSI is stopped, MSIRDY flag goes low after 6 MSI oscillator clock cycles.

T __HAL_RCC_MSI_CALIBRATIONVALUE_ADJUS

Description:

- Macro adjusts Internal Multi Speed oscillator (MSI) calibration value.

Parameters:

- _MSICALIBRATIONVALUE_: specifies the calibration trimming value. (default is RCC_MSICALIBRATION_DEFAULT). This parameter must be a number between 0 and 0xFF.

Notes:

- The calibration is used to compensate for the variations in voltage and temperature that influence the frequency of the internal MSI RC.

__HAL_RCC_MSI_RANGE_CONFIG

__HAL_RCC_GET_MSI_RANGE

Description:

- Macro to get the Internal Multi Speed oscillator (MSI) clock range in run mode.

Return value:

- MSI: clock range. This parameter must be one of the following values:
 - RCC_MSIRANGE_0 MSI clock is around 65.536 KHz
 - RCC_MSIRANGE_1 MSI clock is around 131.072 KHz
 - RCC_MSIRANGE_2 MSI clock is around 262.144 KHz
 - RCC_MSIRANGE_3 MSI clock is around 524.288 KHz
 - RCC_MSIRANGE_4 MSI clock is around 1.048 MHz
 - RCC_MSIRANGE_5 MSI clock is around 2.097 MHz (default after Reset or wake-up from STANDBY)
 - RCC_MSIRANGE_6 MSI clock is around 4.194 MHz

Oscillator Type

RCC_OSCILLATORTYPE_NONE
 RCC_OSCILLATORTYPE_HSE
 RCC_OSCILLATORTYPE_HSI
 RCC_OSCILLATORTYPE_LSE
 RCC_OSCILLATORTYPE_LSI
 RCC_OSCILLATORTYPE_MSI

Peripheral Clock Enable Disable

_HAL_RCC_GPIOA_CLK_ENABLE
 _HAL_RCC_GPIOB_CLK_ENABLE
 _HAL_RCC_GPIOC_CLK_ENABLE
 _HAL_RCC_GPIOD_CLK_ENABLE
 _HAL_RCC_GPIOH_CLK_ENABLE
 _HAL_RCC_CRC_CLK_ENABLE
 _HAL_RCC_FLITF_CLK_ENABLE
 _HAL_RCC_DMA1_CLK_ENABLE
 _HAL_RCC_GPIOA_CLK_DISABLE
 _HAL_RCC_GPIOB_CLK_DISABLE
 _HAL_RCC_GPIOC_CLK_DISABLE
 _HAL_RCC_GPIOD_CLK_DISABLE
 _HAL_RCC_GPIOH_CLK_DISABLE

_HAL_RCC_CRC_CLK_DISABLE
_HAL_RCC_FLITF_CLK_DISABLE
_HAL_RCC_DMA1_CLK_DISABLE

RCC Peripheral Clock Force Release

_HAL_RCC_AHB_FORCE_RESET
_HAL_RCC_GPIOA_FORCE_RESET
_HAL_RCC_GPIOB_FORCE_RESET
_HAL_RCC_GPIOC_FORCE_RESET
_HAL_RCC_GPIOD_FORCE_RESET
_HAL_RCC_GPIOH_FORCE_RESET
_HAL_RCC_CRC_FORCE_RESET
_HAL_RCC_FLITF_FORCE_RESET
_HAL_RCC_DMA1_FORCE_RESET
_HAL_RCC_AHB_RELEASE_RESET
_HAL_RCC_GPIOA_RELEASE_RESET
_HAL_RCC_GPIOB_RELEASE_RESET
_HAL_RCC_GPIOC_RELEASE_RESET
_HAL_RCC_GPIOD_RELEASE_RESET
_HAL_RCC_GPIOH_RELEASE_RESET
_HAL_RCC_CRC_RELEASE_RESET
_HAL_RCC_FLITF_RELEASE_RESET
_HAL_RCC_DMA1_RELEASE_RESET

RCC Peripheral Clock Sleep Enable Disable

_HAL_RCC_GPIOA_CLK_SLEEP_ENABLE
_HAL_RCC_GPIOB_CLK_SLEEP_ENABLE
_HAL_RCC_GPIOC_CLK_SLEEP_ENABLE
_HAL_RCC_GPIOD_CLK_SLEEP_ENABLE
_HAL_RCC_GPIOH_CLK_SLEEP_ENABLE
_HAL_RCC_CRC_CLK_SLEEP_ENABLE
_HAL_RCC_FLITF_CLK_SLEEP_ENABLE
_HAL_RCC_DMA1_CLK_SLEEP_ENABLE
_HAL_RCC_GPIOA_CLK_SLEEP_DISABLE
_HAL_RCC_GPIOB_CLK_SLEEP_DISABLE
_HAL_RCC_GPIOC_CLK_SLEEP_DISABLE
_HAL_RCC_GPIOD_CLK_SLEEP_DISABLE
_HAL_RCC_GPIOH_CLK_SLEEP_DISABLE

```
_HAL_RCC_CRC_CLK_SLEEP_DISABLE  
_HAL_RCC_FLITF_CLK_SLEEP_DISABLE  
_HAL_RCC_DMA1_CLK_SLEEP_DISABLE  
_HAL_RCC_TIM2_CLK_SLEEP_ENABLE
```

Notes:

- Peripheral clock gating in SLEEP mode can be used to further reduce power consumption. After wakeup from SLEEP mode, the peripheral clock is enabled again. By default, all peripheral clocks are enabled during SLEEP mode.

```
_HAL_RCC_TIM3_CLK_SLEEP_ENABLE  
_HAL_RCC_TIM4_CLK_SLEEP_ENABLE  
_HAL_RCC_TIM6_CLK_SLEEP_ENABLE  
_HAL_RCC_TIM7_CLK_SLEEP_ENABLE  
_HAL_RCC_WWDG_CLK_SLEEP_ENABLE  
_HAL_RCC_SPI2_CLK_SLEEP_ENABLE  
_HAL_RCC_USART2_CLK_SLEEP_ENABLE  
_HAL_RCC_USART3_CLK_SLEEP_ENABLE  
_HAL_RCC_I2C1_CLK_SLEEP_ENABLE  
_HAL_RCC_I2C2_CLK_SLEEP_ENABLE  
_HAL_RCC_USB_CLK_SLEEP_ENABLE  
_HAL_RCC_PWR_CLK_SLEEP_ENABLE  
_HAL_RCC_DAC_CLK_SLEEP_ENABLE  
_HAL_RCC_COMP_CLK_SLEEP_ENABLE  
_HAL_RCC_TIM2_CLK_SLEEP_DISABLE  
_HAL_RCC_TIM3_CLK_SLEEP_DISABLE  
_HAL_RCC_TIM4_CLK_SLEEP_DISABLE  
_HAL_RCC_TIM6_CLK_SLEEP_DISABLE  
_HAL_RCC_TIM7_CLK_SLEEP_DISABLE  
_HAL_RCC_WWDG_CLK_SLEEP_DISABLE  
_HAL_RCC_SPI2_CLK_SLEEP_DISABLE  
_HAL_RCC_USART2_CLK_SLEEP_DISABLE  
_HAL_RCC_USART3_CLK_SLEEP_DISABLE  
_HAL_RCC_I2C1_CLK_SLEEP_DISABLE  
_HAL_RCC_I2C2_CLK_SLEEP_DISABLE  
_HAL_RCC_USB_CLK_SLEEP_DISABLE
```

`_HAL_RCC_PWR_CLK_SLEEP_DISABLE`
`_HAL_RCC_DAC_CLK_SLEEP_DISABLE`
`_HAL_RCC_COMP_CLK_SLEEP_DISABLE`
`_HAL_RCC_SYSCFG_CLK_SLEEP_ENABLE`

Notes:

- Peripheral clock gating in SLEEP mode can be used to further reduce power consumption. After wakeup from SLEEP mode, the peripheral clock is enabled again. By default, all peripheral clocks are enabled during SLEEP mode.

`_HAL_RCC_TIM9_CLK_SLEEP_ENABLE`
`_HAL_RCC_TIM10_CLK_SLEEP_ENABLE`
`_HAL_RCC_TIM11_CLK_SLEEP_ENABLE`
`_HAL_RCC_ADC1_CLK_SLEEP_ENABLE`
`_HAL_RCC_SPI1_CLK_SLEEP_ENABLE`
`_HAL_RCC_USART1_CLK_SLEEP_ENABLE`
`_HAL_RCC_SYSCFG_CLK_SLEEP_DISABLE`
`_HAL_RCC_TIM9_CLK_SLEEP_DISABLE`
`_HAL_RCC_TIM10_CLK_SLEEP_DISABLE`
`_HAL_RCC_TIM11_CLK_SLEEP_DISABLE`
`_HAL_RCC_ADC1_CLK_SLEEP_DISABLE`
`_HAL_RCC_SPI1_CLK_SLEEP_DISABLE`
`_HAL_RCC_USART1_CLK_SLEEP_DISABLE`

PLL Clock Source

`RCC_PLLSOURCE_HSI` HSI clock selected as PLL entry clock source

`RCC_PLLSOURCE_HSE` HSE clock selected as PLL entry clock source

PLL Config

<code>RCC_PLL_NONE</code>	PLL is not configured
<code>RCC_PLL_OFF</code>	PLL deactivation
<code>RCC_PLL_ON</code>	PLL activation

PLL Configuration

`_HAL_RCC_PLL_ENABLE`

Notes:

- After enabling the main PLL, the application software should wait on PLLRDY flag to be set indicating that PLL clock is stable and can be used as system clock source. The main PLL is disabled by hardware when entering STOP and STANDBY modes.

[__HAL_RCC_PLL_DISABLE](#)**Notes:**

- The main PLL can not be disabled if it is used as system clock source

[__HAL_RCC_PLL_CONFIG](#)**Description:**

- Macro to configure the main PLL clock source, multiplication and division factors.

Parameters:

- RCC_PLLSOURCE: specifies the PLL entry clock source. This parameter can be one of the following values:
 - RCC_PLLSOURCE_HSI HSI oscillator clock selected as PLL clock entry
 - RCC_PLLSOURCE_HSE HSE oscillator clock selected as PLL clock entry
- PLL_MUL: specifies the multiplication factor for PLL VCO output clock. This parameter can be one of the following values:
 - RCC_PLL_MUL3 PLLVCO = PLL clock entry x 3
 - RCC_PLL_MUL4 PLLVCO = PLL clock entry x 4
 - RCC_PLL_MUL6 PLLVCO = PLL clock entry x 6
 - RCC_PLL_MUL8 PLLVCO = PLL clock entry x 8
 - RCC_PLL_MUL12 PLLVCO = PLL clock entry x 12
 - RCC_PLL_MUL16 PLLVCO = PLL clock entry x 16
 - RCC_PLL_MUL24 PLLVCO = PLL clock entry x 24
 - RCC_PLL_MUL32 PLLVCO = PLL clock entry x 32
 - RCC_PLL_MUL48 PLLVCO = PLL clock entry x 48
- PLL_DIV: specifies the division factor for PLL VCO input clock. This parameter can be one of the following values:
 - RCC_PLL_DIV2 PLL clock output = PLLVCO / 2
 - RCC_PLL_DIV3 PLL clock output = PLLVCO / 3
 - RCC_PLL_DIV4 PLL clock output = PLLVCO / 4

Notes:

- This function must be used only when the

- main PLL is disabled.
- The PLL VCO clock frequency must not exceed 96 MHz when the product is in Range 1, 48 MHz when the product is in Range 2 and 24 MHz when the product is in Range 3.

`_HAL_RCC_GET_PLL_OSCSOURCE`**Description:**

- Get oscillator clock selected as PLL input clock.

Return value:

- The: clock source used for PLL entry. The returned value can be one of the following:
 - RCC_PLLSOURCE_HSI HSI oscillator clock selected as PLL input clock
 - RCC_PLLSOURCE_HSE HSE oscillator clock selected as PLL input clock

PLL Division Factor`RCC_PLL_DIV2``RCC_PLL_DIV3``RCC_PLL_DIV4`***PLL Multiplication Factor***`RCC_PLL_MUL3``RCC_PLL_MUL4``RCC_PLL_MUL6``RCC_PLL_MUL8``RCC_PLL_MUL12``RCC_PLL_MUL16``RCC_PLL_MUL24``RCC_PLL_MUL32``RCC_PLL_MUL48`***Register offsets***`RCC_OFFSET``RCC_CR_OFFSET``RCC_CFGR_OFFSET``RCC_CIR_OFFSET``RCC_CSR_OFFSET`***RCC RTC Clock Configuration***`_HAL_RCC_RTC_CLKPRESCALER`**Description:**

- Macro to configure the RTC clock (RTCCLK).

Parameters:

- `__RTC_CLKSOURCE__`: specifies the RTC clock source. This parameter can be one of the following values:
 - `RCC_RTCCLKSOURCE_NO_CLK`
No clock selected as RTC clock
 - `RCC_RTCCLKSOURCE_LSE` LSE selected as RTC clock
 - `RCC_RTCCLKSOURCE_LSI` LSI selected as RTC clock
 - `RCC_RTCCLKSOURCE_HSE_DIV2`
HSE divided by 2 selected as RTC clock
 - `RCC_RTCCLKSOURCE_HSE_DIV4`
HSE divided by 4 selected as RTC clock
 - `RCC_RTCCLKSOURCE_HSE_DIV8`
HSE divided by 8 selected as RTC clock
 - `RCC_RTCCLKSOURCE_HSE_DIV1`
6 HSE divided by 16 selected as RTC clock

Notes:

- As the RTC clock configuration bits are in the Backup domain and write access is denied to this domain after reset, you have to enable write access using the Power Backup Access macro before to configure the RTC clock source (to be done once after reset). Once the RTC clock is configured it cannot be changed unless the Backup domain is reset using `__HAL_RCC_BACKUPRESET_FORCE()` macro, or by a Power On Reset (POR). RTC prescaler cannot be modified if HSE is enabled (HSEON = 1).
- If the LSE or LSI is used as RTC clock source, the RTC continues to work in STOP and STANDBY modes, and can be used as wakeup source. However, when the HSE clock is used as RTC clock source, the RTC cannot be used in STOP and STANDBY modes. The maximum input clock frequency for RTC is 1MHz (when using HSE as RTC clock source).

`__HAL_RCC_RTC_CONFIG`
`__HAL_RCC_GET_RTC_SOURCE`

Description:

- Macro to get the RTC clock source.

Return value:

- The: clock source can be one of the following values:



- RCC_RTCCLKSOURCE_NO_CLK
No clock selected as RTC clock
- RCC_RTCCLKSOURCE_LSE LSE selected as RTC clock
- RCC_RTCCLKSOURCE_LSI LSI selected as RTC clock
- RCC_RTCCLKSOURCE_HSE_DIVX HSE divided by X selected as RTC clock (X can be retrieved thanks to __HAL_RCC_GET_RTC_HSE_PRESCALER())

`__HAL_RCC_GET_RTC_HSE_PRESCALER`

Description:

- Get the RTC and LCD HSE clock divider (RTCCLK / LCDCLK).

Return value:

- Returned: value can be one of the following values:
 - RCC_RTC_HSE_DIV_2 HSE divided by 2 selected as RTC clock
 - RCC_RTC_HSE_DIV_4 HSE divided by 4 selected as RTC clock
 - RCC_RTC_HSE_DIV_8 HSE divided by 8 selected as RTC clock
 - RCC_RTC_HSE_DIV_16 HSE divided by 16 selected as RTC clock

`__HAL_RCC_RTC_ENABLE`

Notes:

- These macros must be used only after the RTC clock source was selected.

`__HAL_RCC_RTC_DISABLE`

Notes:

- These macros must be used only after the RTC clock source was selected.

`__HAL_RCC_BACKUPRESET_FORCE`

Notes:

- This function resets the RTC peripheral (including the backup registers) and the RTC clock source selection in RCC_CSR register. The BKPSRAM is not affected by this reset.

`__HAL_RCC_BACKUPRESET_RELEASE`

RTC LCD Clock Source

<code>RCC_RTCCLKSOURCE_NO_CLK</code>	No clock
--------------------------------------	----------

<code>RCC_RTCCLKSOURCE_LSE</code>	LSE oscillator clock used as RTC clock
-----------------------------------	--

RCC_RTCCLKSOURCE_LSI	LSI oscillator clock used as RTC clock
RCC_RTCCLKSOURCE_HSE_DIVX	HSE oscillator clock divided by X used as RTC clock
RCC_RTCCLKSOURCE_HSE_DIV2	HSE oscillator clock divided by 2 used as RTC clock
RCC_RTCCLKSOURCE_HSE_DIV4	HSE oscillator clock divided by 4 used as RTC clock
RCC_RTCCLKSOURCE_HSE_DIV8	HSE oscillator clock divided by 8 used as RTC clock
RCC_RTCCLKSOURCE_HSE_DIV16	HSE oscillator clock divided by 16 used as RTC clock

System Clock Source

RCC_SYSCLKSOURCE_MSI	MSI selected as system clock
RCC_SYSCLKSOURCE_HSI	HSI selected as system clock
RCC_SYSCLKSOURCE_HSE	HSE selected as system clock
RCC_SYSCLKSOURCE_PLLCLK	PLL selected as system clock

System Clock Source Status

RCC_SYSCLKSOURCE_STATUS_MSI	MSI used as system clock
RCC_SYSCLKSOURCE_STATUS_HSI	HSI used as system clock
RCC_SYSCLKSOURCE_STATUS_HSE	HSE used as system clock
RCC_SYSCLKSOURCE_STATUS_PLLCLK	PLL used as system clock

System Clock Type

RCC_CLOCKTYPE_SYSCLK	SYSCLK to configure
RCC_CLOCKTYPE_HCLK	HCLK to configure
RCC_CLOCKTYPE_PCLK1	PCLK1 to configure
RCC_CLOCKTYPE_PCLK2	PCLK2 to configure

RCC Timeout

RCC_DBP_TIMEOUT_VALUE	
RCC_LSE_TIMEOUT_VALUE	
CLOCKSWITCH_TIMEOUT_VALUE	
HSE_TIMEOUT_VALUE	
MSI_TIMEOUT_VALUE	
HSI_TIMEOUT_VALUE	
LSI_TIMEOUT_VALUE	
PLL_TIMEOUT_VALUE	

35 HAL RCC Extension Driver

35.1 RCCEEx Firmware driver registers structures

35.1.1 RCC_PeriphCLKInitTypeDef

Data Fields

- *uint32_t PeriphClockSelection*
- *uint32_t RTCClockSelection*
- *uint32_t LCDClockSelection*

Field Documentation

- *uint32_t RCC_PeriphCLKInitTypeDef::PeriphClockSelection*
The Extended Clock to be configured. This parameter can be a value of [RCCEx_Periph_Clock_Selection](#)
- *uint32_t RCC_PeriphCLKInitTypeDef::RTCClockSelection*
specifies the RTC clock source. This parameter can be a value of [RCC_RTC_LCD_Clock_Source](#)
- *uint32_t RCC_PeriphCLKInitTypeDef::LCDClockSelection*
specifies the LCD clock source. This parameter can be a value of [RCC_RTC_LCD_Clock_Source](#)

35.2 RCCEEx Firmware driver API description

35.2.1 Extended Peripheral Control functions

This subsection provides a set of functions allowing to control the RCC Clocks frequencies.



Important note: Care must be taken when `HAL_RCCEEx_PeriphCLKConfig()` is used to select the RTC clock source; in this case the Backup domain will be reset in order to modify the RTC Clock source, as consequence RTC registers (including the backup registers) are set to their reset values.

This section contains the following APIs:

- [HAL_RCCEEx_PeriphCLKConfig\(\)](#)
- [HAL_RCCEEx_GetPeriphCLKConfig\(\)](#)
- [HAL_RCCEEx_GetPeriphCLKFreq\(\)](#)
- [HAL_RCCEEx_EnableLSECSS\(\)](#)
- [HAL_RCCEEx_DisableLSECSS\(\)](#)
- [HAL_RCCEEx_EnableLSECSS_IT\(\)](#)
- [HAL_RCCEEx_LSECSS_IRQHandler\(\)](#)
- [HAL_RCCEEx_LSECSS_Callback\(\)](#)

35.2.2 Detailed description of functions

`HAL_RCCEEx_PeriphCLKConfig`

Function name `HAL_StatusTypeDef HAL_RCCEEx_PeriphCLKConfig
(RCC_PeriphCLKInitTypeDef * PeriphClkInit)`

Function description	Initializes the RCC extended peripherals clocks according to the specified parameters in the RCC_PeriphCLKInitTypeDef.
Parameters	<ul style="list-style-type: none"> PeriphClkInit: pointer to an RCC_PeriphCLKInitTypeDef structure that contains the configuration information for the Extended Peripherals clocks(RTC/LCD clock).
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> If HAL_ERROR returned, first switch-OFF HSE clock oscillator with HAL_RCC_OscConfig() to possibly update HSE divider.

HAL_RCCEEx_GetPeriphCLKConfig

Function name	<code>void HAL_RCCEEx_GetPeriphCLKConfig (RCC_PeriphCLKInitTypeDef * PeriphClkInit)</code>
Function description	Get the PeriphClkInit according to the internal RCC configuration registers.
Parameters	<ul style="list-style-type: none"> PeriphClkInit: pointer to an RCC_PeriphCLKInitTypeDef structure that returns the configuration information for the Extended Peripherals clocks(RTC/LCD clocks).
Return values	<ul style="list-style-type: none"> None:

HAL_RCCEEx_GetPeriphCLKFreq

Function name	<code>uint32_t HAL_RCCEEx_GetPeriphCLKFreq (uint32_t PeriphClk)</code>
Function description	Return the peripheral clock frequency.
Parameters	<ul style="list-style-type: none"> PeriphClk: Peripheral clock identifier This parameter can be one of the following values: <ul style="list-style-type: none"> RCC_PERIPHCLK_RTC RTC peripheral clock RCC_PERIPHCLK_LCD LCD peripheral clock (*)
Return values	<ul style="list-style-type: none"> Frequency: in Hz (0: means that no available frequency for the peripheral)
Notes	<ul style="list-style-type: none"> Return 0 if peripheral clock is unknown (*) means that this peripheral is not present on all the devices

HAL_RCCEEx_EnableLSECSS

Function name	<code>void HAL_RCCEEx_EnableLSECSS (void)</code>
Function description	Enables the LSE Clock Security System.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC but no hardware action is made to the registers. In Standby mode a wakeup is generated. In other modes an interrupt can be sent to wakeup the software (see Section 5.3.4: Clock interrupt register (RCC_CIR) on page 104). The software MUST then disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON), and can change the RTC clock source

(no clock or LSI or HSE, with RTCSEL), or take any required action to secure the application.

- LSE CSS available only for high density and medium+ devices

HAL_RCCEEx_DisableLSECSS

Function name	void HAL_RCCEEx_DisableLSECSS (void)
Function description	Disables the LSE Clock Security System.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Once enabled this bit cannot be disabled, except after an LSE failure detection (LSECSSD=1). In that case the software MUST disable the LSECSSON bit. Reset by power on reset and RTC software reset (RTCRST bit). • LSE CSS available only for high density and medium+ devices

HAL_RCCEEx_EnableLSECSS_IT

Function name	void HAL_RCCEEx_EnableLSECSS_IT (void)
Function description	Enable the LSE Clock Security System IT & corresponding EXTI line.
Return values	<ul style="list-style-type: none"> • None:
Notes	LSE Clock Security System IT is mapped on RTC EXTI line 19

HAL_RCCEEx_LSECSS_IRQHandler

Function name	void HAL_RCCEEx_LSECSS_IRQHandler (void)
Function description	Handle the RCC LSE Clock Security System interrupt request.
Return values	<ul style="list-style-type: none"> • None:

HAL_RCCEEx_LSECSS_Callback

Function name	void HAL_RCCEEx_LSECSS_Callback (void)
Function description	RCCEx LSE Clock Security System interrupt callback.
Return values	<ul style="list-style-type: none"> • none:

35.3 RCCEx Firmware driver defines

35.3.1 RCCEx

RCCEx Exported Macros

_HAL_RCC_LSECSS_EXTI_ENABLE_IT	Description:
	<ul style="list-style-type: none"> • Enable interrupt on RCC LSE CSS EXTI Line 19.

`__HAL_RCC_LSECSS_EXTI_DISABLE_IT`

Return value:

- None

`__HAL_RCC_LSECSS_EXTI_ENABLE_EVENT`

Return value:

- None

`__HAL_RCC_LSECSS_EXTI_DISABLE_EVENT`

Return value:

- None.

`__HAL_RCC_LSECSS_EXTI_ENABLE_FALLING_EDGE`

Return value:

- None.

`__HAL_RCC_LSECSS_EXTI_DISABLE_FALLING_EDGE`

Description:

- RCC LSE CSS EXTI line configuration: set falling edge trigger.

Return value:

- None.

`__HAL_RCC_LSECSS_EXTI_ENABLE_RISING_EDGE`

Description:

- Disable the RCC LSE CSS Extended Interrupt Falling Trigger.

Return value:

- None.

`__HAL_RCC_LSECSS_EXTI_DISABLE_RISING_EDGE`

Description:

- RCC LSE CSS EXTI line configuration: set rising edge trigger.

Return value:

- None.

Description:

- Disable the RCC LSE CSS Extended Interrupt Rising Trigger.

Return value:

- None.

`__HAL_RCC_LSECSS_EXTI_ENABLE_RISING_
FALLING_EDGE`

Description:

- RCC LSE CSS EXTI line configuration: set rising & falling edge trigger.

Return value:

- None.

`__HAL_RCC_LSECSS_EXTI_DISABLE_RISING_
FALLING_EDGE`

Description:

- Disable the RCC LSE CSS Extended Interrupt Rising & Falling Trigger.

Return value:

- None.

`__HAL_RCC_LSECSS_EXTI_GET_FLAG`

Description:

- Check whether the specified RCC LSE CSS EXTI interrupt flag is set or not.

Return value:

- EXTI: RCC LSE CSS Line Status.

`__HAL_RCC_LSECSS_EXTI_CLEAR_FLAG`

Description:

- Clear the RCC LSE CSS EXTI flag.

Return value:

- None.

`__HAL_RCC_LSECSS_EXTI_GENERATE_SWIT`

Description:

- Generate a Software interrupt on selected EXTI line.

Return value:

- None.

RCC LSE CSS external interrupt line

`RCC_EXTI_LINE_LSECSS` External interrupt line 19 connected to the LSE CSS EXTI Line

RCCEx Force Release Peripheral Reset

`__HAL_RCC_GPIOE_FORCE_RESET`
`__HAL_RCC_GPIOE_RELEASE_RESET`
`__HAL_RCC_GPIOF_FORCE_RESET`
`__HAL_RCC_GPIOG_FORCE_RESET`
`__HAL_RCC_GPIOF_RELEASE_RESET`

```
_HAL_RCC_GPIOG_RELEASE_RESET  
_HAL_RCC_DMA2_FORCE_RESET  
_HAL_RCC_DMA2_RELEASE_RESET  
_HAL_RCC_CRYPT_FORCE_RESET  
_HAL_RCC_CRYPT_RELEASE_RESET  
_HAL_RCC_FSMC_FORCE_RESET  
_HAL_RCC_FSMC_RELEASE_RESET  
_HAL_RCC_LCD_FORCE_RESET  
_HAL_RCC_LCD_RELEASE_RESET  
_HAL_RCC_TIM5_FORCE_RESET  
_HAL_RCC_TIM5_RELEASE_RESET  
_HAL_RCC_SPI3_FORCE_RESET  
_HAL_RCC_SPI3_RELEASE_RESET  
_HAL_RCC_UART4_FORCE_RESET  
_HAL_RCC_UART5_FORCE_RESET  
_HAL_RCC_UART4_RELEASE_RESET  
_HAL_RCC_UART5_RELEASE_RESET  
_HAL_RCC_OPAMP_FORCE_RESET  
_HAL_RCC_OPAMP_RELEASE_RESET  
_HAL_RCC_SDIO_FORCE_RESET  
_HAL_RCC_SDIO_RELEASE_RESET
```

LCD Configuration

```
_HAL_RCC_LCD_CONFIG
```

Description:

- Macro to configures LCD clock (LCDCLK).

Parameters:

- LCD_CLKSOURCE: specifies the LCD clock source. This parameter can be one of the following values:
 - RCC_RTCCLKSOURCE_LSE LSE selected as LCD clock
 - RCC_RTCCLKSOURCE_LSI LSI selected as LCD clock
 - RCC_RTCCLKSOURCE_HSE_DIV2 HSE divided by 2 selected as LCD clock
 - RCC_RTCCLKSOURCE_HSE_DIV4 HSE divided by 4 selected as LCD clock
 - RCC_RTCCLKSOURCE_HSE_DIV8 HSE divided by 8 selected as LCD clock
 - RCC_RTCCLKSOURCE_HSE_DIV1

6 HSE divided by 16 selected as LCD clock

Notes:

- LCD and RTC use the same configuration
LCD can however be used in the Stop low power mode if the LSE or LSI is used as the LCD clock source.

```
__HAL_RCC_GET_LCD_SOURCE
__HAL_RCC_GET_LCD_HSE_PRESC
ALER
```

RCC Extended MCOx Clock Config

`__HAL_RCC_MCO1_CONFIG` **Description:**

- Macro to configure the MCO clock.

Parameters:

- `__MCOCLKSOURCE__`: specifies the MCO clock source. This parameter can be one of the following values:
 - `RCC_MCO1SOURCE_NOCLOCK` No clock selected as MCO clock
 - `RCC_MCO1SOURCE_SYSCLK` System Clock selected as MCO clock
 - `RCC_MCO1SOURCE_HSI` HSI oscillator clock selected as MCO clock
 - `RCC_MCO1SOURCE_MSI` MSI oscillator clock selected as MCO clock
 - `RCC_MCO1SOURCE_HSE` HSE oscillator clock selected as MCO clock
 - `RCC_MCO1SOURCE_PLLCLK` PLL clock selected as MCO clock
 - `RCC_MCO1SOURCE_LSI` LSI clock selected as MCO clock
 - `RCC_MCO1SOURCE_LSE` LSE clock selected as MCO clock
- `__MCODIV__`: specifies the MCO clock prescaler. This parameter can be one of the following values:
 - `RCC_MCODIV_1` MCO clock source is divided by 1
 - `RCC_MCODIV_2` MCO clock source is divided by 2
 - `RCC_MCODIV_4` MCO clock source is divided by 4
 - `RCC_MCODIV_8` MCO clock source is divided by 8
 - `RCC_MCODIV_16` MCO clock source is divided by 16

RCCEx_Peripheral_Clock_Enable_Disable

```
__HAL_RCC_GPIOE_CLK_ENABLE
__HAL_RCC_GPIOE_CLK_DISABLE
```

```
_HAL_RCC_GPIOF_CLK_ENABLE  
_HAL_RCC_GPIOG_CLK_ENABLE  
_HAL_RCC_GPIOF_CLK_DISABLE  
_HAL_RCC_GPIOG_CLK_DISABLE  
_HAL_RCC_DMA2_CLK_ENABLE  
_HAL_RCC_DMA2_CLK_DISABLE  
_HAL_RCC_CRYP_CLK_ENABLE  
_HAL_RCC_CRYP_CLK_DISABLE  
_HAL_RCC_FSMC_CLK_ENABLE  
_HAL_RCC_FSMC_CLK_DISABLE  
_HAL_RCC_LCD_CLK_ENABLE  
_HAL_RCC_LCD_CLK_DISABLE  
_HAL_RCC_TIM5_CLK_ENABLE
```

Notes:

- After reset, the peripheral clock (used for registers read/write access) is disabled and the application software has to enable this clock before using it.

```
_HAL_RCC_TIM5_CLK_DISABLE  
_HAL_RCC_SPI3_CLK_ENABLE  
_HAL_RCC_SPI3_CLK_DISABLE  
_HAL_RCC_UART4_CLK_ENABLE  
_HAL_RCC_UART5_CLK_ENABLE  
_HAL_RCC_UART4_CLK_DISABLE  
_HAL_RCC_UART5_CLK_DISABLE  
_HAL_RCC_OPAMP_CLK_ENABLE  
_HAL_RCC_OPAMP_CLK_DISABLE  
_HAL_RCC_SDIO_CLK_ENABLE
```

Notes:

- After reset, the peripheral clock (used for registers read/write access) is disabled and the application software has to enable this clock before using it.

```
_HAL_RCC_SDIO_CLK_DISABLE
```

Peripheral Clock Enable Disable Status

```
_HAL_RCC_GPIOE_IS_CLK_ENABLED  
_HAL_RCC_GPIOE_IS_CLK_DISABLED  
_HAL_RCC_GPIOF_IS_CLK_ENABLED
```

```
_HAL_RCC_GPIOG_IS_CLK_ENABLED  
_HAL_RCC_GPIOF_IS_CLK_DISABLED  
_HAL_RCC_GPIOG_IS_CLK_DISABLED  
_HAL_RCC_DMA2_IS_CLK_ENABLED  
_HAL_RCC_DMA2_IS_CLK_DISABLED  
_HAL_RCC_CRYP_IS_CLK_ENABLED  
_HAL_RCC_CRYP_IS_CLK_DISABLED  
_HAL_RCC_FSMC_IS_CLK_ENABLED  
_HAL_RCC_FSMC_IS_CLK_DISABLED  
_HAL_RCC_LCD_IS_CLK_ENABLED  
_HAL_RCC_LCD_IS_CLK_DISABLED  
_HAL_RCC_TIM5_IS_CLK_ENABLED  
_HAL_RCC_TIM5_IS_CLK_DISABLED  
_HAL_RCC_SPI3_IS_CLK_ENABLED  
_HAL_RCC_SPI3_IS_CLK_DISABLED  
_HAL_RCC_UART4_IS_CLK_ENABLED  
_HAL_RCC_UART5_IS_CLK_ENABLED  
_HAL_RCC_UART4_IS_CLK_DISABLED  
_HAL_RCC_UART5_IS_CLK_DISABLED  
_HAL_RCC_OPAMP_IS_CLK_ENABLED  
_HAL_RCC_OPAMP_IS_CLK_DISABLED  
_HAL_RCC_SDIO_IS_CLK_ENABLED  
_HAL_RCC_SDIO_IS_CLK_DISABLED
```

RCCEx Peripheral Clock Sleep Enable Disable

```
_HAL_RCC_GPIOE_CLK_SLEEP_ENABLE  
_HAL_RCC_GPIOE_CLK_SLEEP_DISABLE  
_HAL_RCC_GPIOF_CLK_SLEEP_ENABLE  
_HAL_RCC_GPIOG_CLK_SLEEP_ENABLE  
_HAL_RCC_GPIOF_CLK_SLEEP_DISABLE  
_HAL_RCC_GPIOG_CLK_SLEEP_DISABLE  
_HAL_RCC_DMA2_CLK_SLEEP_ENABLE  
_HAL_RCC_DMA2_CLK_SLEEP_DISABLE  
_HAL_RCC_CRYP_CLK_SLEEP_ENABLE  
_HAL_RCC_CRYP_CLK_SLEEP_DISABLE  
_HAL_RCC_FSMC_CLK_SLEEP_ENABLE  
_HAL_RCC_FSMC_CLK_SLEEP_DISABLE
```

```
_HAL_RCC_LCD_CLK_SLEEP_ENABLE  
_HAL_RCC_LCD_CLK_SLEEP_DISABLE  
_HAL_RCC_TIM5_CLK_SLEEP_ENABLE
```

Notes:

- Peripheral clock gating in SLEEP mode can be used to further reduce power consumption. After wakeup from SLEEP mode, the peripheral clock is enabled again. By default, all peripheral clocks are enabled during SLEEP mode.

```
_HAL_RCC_TIM5_CLK_SLEEP_DISABLE  
_HAL_RCC_SPI3_CLK_SLEEP_ENABLE  
_HAL_RCC_SPI3_CLK_SLEEP_DISABLE  
_HAL_RCC_UART4_CLK_SLEEP_ENABLE  
_HAL_RCC_UART5_CLK_SLEEP_ENABLE  
_HAL_RCC_UART4_CLK_SLEEP_DISABLE  
_HAL_RCC_UART5_CLK_SLEEP_DISABLE  
_HAL_RCC_OPAMP_CLK_SLEEP_ENABLE  
_HAL_RCC_OPAMP_CLK_SLEEP_DISABLE  
_HAL_RCC_SDIO_CLK_SLEEP_ENABLE
```

Notes:

- Peripheral clock gating in SLEEP mode can be used to further reduce power consumption. After wakeup from SLEEP mode, the peripheral clock is enabled again. By default, all peripheral clocks are enabled during SLEEP mode.

```
_HAL_RCC_SDIO_CLK_SLEEP_DISABLE
```

Peripheral Clock Sleep Enable Disable Status

```
_HAL_RCC_GPIOE_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_GPIOE_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_GPIOF_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_GPIOG_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_GPIOF_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_GPIOG_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_DMA2_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_DMA2_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_CRYP_IS_CLK_SLEEP_ENABLED
```

```
_HAL_RCC_CRYP_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_FSMC_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_FSMC_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_LCD_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_LCD_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_TIM5_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_TIM5_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_SPI3_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_SPI3_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_UART4_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_UART5_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_UART4_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_UART5_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_OPAMP_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_OPAMP_IS_CLK_SLEEP_DISABLED  
_HAL_RCC_SDIO_IS_CLK_SLEEP_ENABLED  
_HAL_RCC_SDIO_IS_CLK_SLEEP_DISABLED
```

RCCEx Periph Clock Selection

RCC_PERIPHCLK_RTC

RCC_PERIPHCLK_LCD

36 HAL RTC Generic Driver

36.1 RTC Firmware driver registers structures

36.1.1 RTC_InitTypeDef

Data Fields

- *uint32_t HourFormat*
- *uint32_t AsynchPrediv*
- *uint32_t SynchPrediv*
- *uint32_t OutPut*
- *uint32_t OutPutPolarity*
- *uint32_t OutPutType*

Field Documentation

- *uint32_t RTC_InitTypeDef::HourFormat*
Specifies the RTC Hour Format. This parameter can be a value of
[*RTC_Hour_Formats*](#)
- *uint32_t RTC_InitTypeDef::AsynchPrediv*
Specifies the RTC Asynchronous Predivider value. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x7F
- *uint32_t RTC_InitTypeDef::SynchPrediv*
Specifies the RTC Synchronous Predivider value. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFFFF
- *uint32_t RTC_InitTypeDef::OutPut*
Specifies which signal will be routed to the RTC output. This parameter can be a value of
[*RTCEx_Output_selection_Definitions*](#)
- *uint32_t RTC_InitTypeDef::OutPutPolarity*
Specifies the polarity of the output signal. This parameter can be a value of
[*RTC_Output_Polarity_Definitions*](#)
- *uint32_t RTC_InitTypeDef::OutPutType*
Specifies the RTC Output Pin mode. This parameter can be a value of
[*RTC_Output_Type_ALARM_OUT*](#)

36.1.2 RTC_DateTypeDef

Data Fields

- *uint8_t WeekDay*
- *uint8_t Month*
- *uint8_t Date*
- *uint8_t Year*

Field Documentation

- *uint8_t RTC_DateTypeDef::WeekDay*
Specifies the RTC Date WeekDay. This parameter can be a value of
[*RTC_WeekDay_Definitions*](#)
- *uint8_t RTC_DateTypeDef::Month*
Specifies the RTC Date Month (in BCD format). This parameter can be a value of
[*RTC_Month_Date_Definitions*](#)

- ***uint8_t RTC_DateTypeDef::Date***
Specifies the RTC Date. This parameter must be a number between Min_Data = 1 and Max_Data = 31
- ***uint8_t RTC_DateTypeDef::Year***
Specifies the RTC Date Year. This parameter must be a number between Min_Data = 0 and Max_Data = 99

36.1.3 RTC_HandleTypeDef

Data Fields

- ***RTC_TypeDef * Instance***
- ***RTC_InitTypeDef Init***
- ***HAL_LockTypeDef Lock***
- ***__IO HAL_RTCStateTypeDef State***

Field Documentation

- ***RTC_TypeDef* RTC_HandleTypeDef::Instance***
Register base address
- ***RTC_InitTypeDef RTC_HandleTypeDef::Init***
RTC required parameters
- ***HAL_LockTypeDef RTC_HandleTypeDef::Lock***
RTC locking object
- ***__IO HAL_RTCStateTypeDef RTC_HandleTypeDef::State***
Time communication state

36.2 RTC Firmware driver API description

36.2.1 Backup Domain Operating Condition

The real-time clock (RTC) and the RTC backup registers can be powered from the VBAT voltage when the main VDD supply is powered off. To retain the content of the RTC backup registers and supply the RTC when VDD is turned off, VBAT pin can be connected to an optional standby voltage supplied by a battery or by another source.

To allow the RTC operating even when the main digital supply (VDD) is turned off, the VBAT pin powers the following blocks:

1. The RTC
2. The LSE oscillator
3. PC13 to PC15 I/Os (when available)

When the backup domain is supplied by VDD (analog switch connected to VDD), the following pins are available:

1. PC14 and PC15 can be used as either GPIO or LSE pins
2. PC13 can be used as a GPIO or as the RTC_AF1 pin

When the backup domain is supplied by VBAT (analog switch connected to VBAT because VDD is not present), the following pins are available:

1. PC14 and PC15 can be used as LSE pins only
2. PC13 can be used as the RTC_AF1 pin

36.2.2 Backup Domain Reset

The backup domain reset sets all RTC registers and the RCC_BDCR register to their reset values.

A backup domain reset is generated when one of the following events occurs:

1. Software reset, triggered by setting the BDRST bit in the RCC Backup domain control register (RCC_BDCR).
2. VDD or VBAT power on, if both supplies have previously been powered off.

36.2.3 Backup Domain Access

After reset, the backup domain (RTC registers, RTC backup data registers and backup SRAM) is protected against possible unwanted write accesses.

To enable access to the RTC Domain and RTC registers, proceed as follows:

- Enable the Power Controller (PWR) APB1 interface clock using the __HAL_RCC_PWR_CLK_ENABLE() function.
- Enable access to RTC domain using the HAL_PWR_EnableBkUpAccess() function.
- Select the RTC clock source using the __HAL_RCC_RTC_CONFIG() function.
- Enable RTC Clock using the __HAL_RCC_RTC_ENABLE() function.

36.2.4 How to use this driver

- Enable the RTC domain access (see description in the section above).
- Configure the RTC Prescaler (Asynchronous and Synchronous) and RTC hour format using the HAL_RTC_Init() function.

Time and Date configuration

- To configure the RTC Calendar (Time and Date) use the HAL_RTC_SetTime() and HAL_RTC_SetDate() functions.
- To read the RTC Calendar, use the HAL_RTC_GetTime() and HAL_RTC_GetDate() functions.

Alarm configuration

- To configure the RTC Alarm use the HAL_RTC_SetAlarm() function. You can also configure the RTC Alarm with interrupt mode using the HAL_RTC_SetAlarm_IT() function.
- To read the RTC Alarm, use the HAL_RTC_GetAlarm() function.

36.2.5 RTC and low power modes

The MCU can be woken up from a low power mode by an RTC alternate function.

The RTC alternate functions are the RTC alarms (Alarm A and Alarm B), RTC wakeup, RTC tamper event detection and RTC time stamp event detection. These RTC alternate functions can wake up the system from the Stop and Standby low power modes.

The system can also wake up from low power modes without depending on an external interrupt (Auto-wakeup mode), by using the RTC alarm or the RTC wakeup events.

The RTC provides a programmable time base for waking up from the Stop or Standby mode at regular intervals. Wakeup from STOP and STANDBY modes is possible only when the RTC clock source is LSE or LSI.

36.2.6 Initialization and de-initialization functions

This section provides functions allowing to initialize and configure the RTC Prescaler (Synchronous and Asynchronous), RTC Hour format, disable RTC registers Write protection, enter and exit the RTC initialization mode, RTC registers synchronization check and reference clock detection enable.

1. The RTC Prescaler is programmed to generate the RTC 1Hz time base. It is split into 2 programmable prescalers to minimize power consumption.
 - A 7-bit asynchronous prescaler and a 13-bit synchronous prescaler.
 - When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize power consumption.
2. All RTC registers are Write protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.
3. To configure the RTC Calendar, user application should enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated. When the initialization sequence is complete, the calendar restarts counting after 4 RTCCLK cycles.
4. To read the calendar through the shadow registers after Calendar initialization, calendar update or after wakeup from low power modes the software must first clear the RSF flag. The software must then wait until it is set again before reading the calendar, which means that the calendar registers have been correctly copied into the RTC_TR and RTC_DR shadow registers. The HAL_RTC_WaitForSynchro() function implements the above software sequence (RSF clear and RSF check).

This section contains the following APIs:

- [*HAL_RTC_Init\(\)*](#)
- [*HAL_RTC_DelInit\(\)*](#)
- [*HAL_RTC_MspInit\(\)*](#)
- [*HAL_RTC_MspDelInit\(\)*](#)
- [*HAL_RTC_SetTime\(\)*](#)
- [*HAL_RTC_GetTime\(\)*](#)
- [*HAL_RTC_SetDate\(\)*](#)
- [*HAL_RTC_GetDate\(\)*](#)

36.2.7 RTC Time and Date functions

This section provides functions allowing to configure Time and Date features

This section contains the following APIs:

- [*HAL_RTC_SetTime\(\)*](#)
- [*HAL_RTC_SetDate\(\)*](#)
- [*HAL_RTC_GetDate\(\)*](#)
- [*HAL_RTC_GetTime\(\)*](#)
- [*HAL_RTC_SetAlarm\(\)*](#)
- [*HAL_RTC_SetAlarm_IT\(\)*](#)
- [*HAL_RTC_DeactivateAlarm\(\)*](#)
- [*HAL_RTC_GetAlarm\(\)*](#)
- [*HAL_RTC_AlarmIRQHandler\(\)*](#)
- [*HAL_RTC_PollForAlarmAEvent\(\)*](#)
- [*HAL_RTC_AlarmAEventCallback\(\)*](#)

36.2.8 RTC Alarm functions

This section provides functions allowing to configure Alarm feature

This section contains the following APIs:

- [*HAL_RTC_DeactivateAlarm\(\)*](#)
- [*HAL_RTC_AlarmIRQHandler\(\)*](#)
- [*HAL_RTC_AlarmAEventCallback\(\)*](#)
- [*HAL_RTC_PollForAlarmAEvent\(\)*](#)
- [*HAL_RTC_SetAlarm\(\)*](#)

- [*HAL_RTC_SetAlarm_IT\(\)*](#)
- [*HAL_RTC_GetAlarm\(\)*](#)
- [*HAL_RTC_WaitForSynchro\(\)*](#)

36.2.9 Peripheral State functions

This subsection provides functions allowing to

- Get RTC state

This section contains the following APIs:

- [*HAL_RTC_GetState\(\)*](#)

36.2.10 Peripheral Control functions

This subsection provides functions allowing to

- Wait for RTC Time and Date Synchronization

This section contains the following APIs:

- [*HAL_RTC_WaitForSynchro\(\)*](#)

36.2.11 Detailed description of functions

HAL_RTC_Init

Function name	HAL_StatusTypeDef HAL_RTC_Init (RTC_HandleTypeDef * hrtc)
Function description	Initializes the RTC peripheral.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTC_DeInit

Function name	HAL_StatusTypeDef HAL_RTC_DeInit (RTC_HandleTypeDef * hrtc)
Function description	DeInitializes the RTC peripheral.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • HAL: status • HAL: status

Notes

- This function doesn't reset the RTC Backup Data registers.
- This function does not reset the RTC Backup Data registers.

HAL_RTC_MspInit

Function name	void HAL_RTC_MspInit (RTC_HandleTypeDef * hrtc)
Function description	Initializes the RTC MSP.

Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTC_MspDelInit

Function name	void HAL_RTC_MspDelInit (RTC_HandleTypeDef * hrtc)
Function description	DeInitializes the RTC MSP.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTC_SetTime

Function name	HAL_StatusTypeDef HAL_RTC_SetTime (RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTime, uint32_t Format)
Function description	Sets RTC current time.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • sTime: Pointer to Time structure • Format: Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> – RTC_FORMAT_BIN: Binary data format – RTC_FORMAT_BCD: BCD data format
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTC_GetTime

Function name	HAL_StatusTypeDef HAL_RTC_GetTime (RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTime, uint32_t Format)
Function description	Get RTC current time.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • sTime: Pointer to Time structure with Hours, Minutes and Seconds fields returned with input format (BIN or BCD), also SubSeconds field (if available) returning the RTC_SSR register content and SecondFraction field the Synchronous pre-scaler factor to be used for second fraction ratio computation. • Format: Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> – RTC_FORMAT_BIN: Binary data format – RTC_FORMAT_BCD: BCD data format
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • If available, you can use SubSeconds and SecondFraction (sTime structure fields returned) to convert SubSeconds value in second fraction ratio with time unit following generic formula: Second fraction ratio * time_unit= [(SecondFraction -

- SubSeconds)/(SecondFraction+1)] * time_unit This conversion can be performed only if no shift operation is pending (ie. SHFP=0) when PREDIV_S >= SS
- You must call HAL_RTC_GetDate() after HAL_RTC_GetTime() to unlock the values in the higher-order calendar shadow registers to ensure consistency between the time and date values. Reading RTC current time locks the values in calendar shadow registers until Current date is read to ensure consistency between the time and date values.

HAL_RTC_SetDate

Function name	HAL_StatusTypeDef HAL_RTC_SetDate (RTC_HandleTypeDef * hrtc, RTC_DateTypeDef * sDate, uint32_t Format)
Function description	Sets RTC current date.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • sDate: Pointer to date structure • Format: specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> – RTC_FORMAT_BIN: Binary data format – RTC_FORMAT_BCD: BCD data format
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTC_GetDate

Function name	HAL_StatusTypeDef HAL_RTC_GetDate (RTC_HandleTypeDef * hrtc, RTC_DateTypeDef * sDate, uint32_t Format)
Function description	Gets RTC current date.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • sDate: Pointer to Date structure • Format: Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> – RTC_FORMAT_BIN: Binary data format – RTC_FORMAT_BCD: BCD data format
Return values	<ul style="list-style-type: none"> • HAL: status

Notes

- You must call HAL_RTC_GetDate() after HAL_RTC_GetTime() to unlock the values in the higher-order calendar shadow registers to ensure consistency between the time and date values. Reading RTC current time locks the values in calendar shadow registers until Current date is read.

HAL_RTC_SetAlarm

Function name	HAL_StatusTypeDef HAL_RTC_SetAlarm (RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm, uint32_t Format)
Function description	Sets the specified RTC Alarm.

Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. sAlarm: Pointer to Alarm structure Format: Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> – RTC_FORMAT_BIN: Binary data format – RTC_FORMAT_BCD: BCD data format
Return values	<ul style="list-style-type: none"> HAL: status

HAL_RTC_SetAlarm_IT

Function name	HAL_StatusTypeDef HAL_RTC_SetAlarm_IT (RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm, uint32_t Format)
Function description	Sets the specified RTC Alarm with Interrupt.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. sAlarm: Pointer to Alarm structure Format: Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> – RTC_FORMAT_BIN: Binary data format – RTC_FORMAT_BCD: BCD data format
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> The Alarm register can only be written when the corresponding Alarm is disabled (Use the HAL_RTC_DeactivateAlarm()). The HAL_RTC_SetTime() must be called before enabling the Alarm feature.

HAL_RTC_DeactivateAlarm

Function name	HAL_StatusTypeDef HAL_RTC_DeactivateAlarm (RTC_HandleTypeDef * hrtc, uint32_t Alarm)
Function description	Deactive the specified RTC Alarm.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. Alarm: Specifies the Alarm. This parameter can be one of the following values: <ul style="list-style-type: none"> – RTC_ALARM_A: AlarmA – RTC_ALARM_B: AlarmB
Return values	<ul style="list-style-type: none"> HAL: status

HAL_RTC_GetAlarm

Function name	HAL_StatusTypeDef HAL_RTC_GetAlarm (RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm, uint32_t Alarm, uint32_t Format)
Function description	Gets the RTC Alarm value and masks.

Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. sAlarm: Pointer to Date structure Alarm: Specifies the Alarm. This parameter can be one of the following values: <ul style="list-style-type: none"> RTC_ALARM_A: AlarmA RTC_ALARM_B: AlarmB Format: Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> RTC_FORMAT_BIN: Binary data format RTC_FORMAT_BCD: BCD data format
Return values	<ul style="list-style-type: none"> HAL: status

HAL_RTC_AlarmIRQHandler

Function name	void HAL_RTC_AlarmIRQHandler (RTC_HandleTypeDef * hrtc)
Function description	This function handles Alarm interrupt request.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> None:

HAL_RTC_PollForAlarmAEvent

Function name	HAL_StatusTypeDef HAL_RTC_PollForAlarmAEvent (RTC_HandleTypeDef * hrtc, uint32_t Timeout)
Function description	This function handles AlarmA Polling request.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status

HAL_RTC_AlarmAEventCallback

Function name	void HAL_RTC_AlarmAEventCallback (RTC_HandleTypeDef * hrtc)
Function description	Alarm A callback.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> None:

HAL_RTC_WaitForSynchro

Function name	HAL_StatusTypeDef HAL_RTC_WaitForSynchro (RTC_HandleTypeDef * hrtc)
Function description	Waits until the RTC Time and Date registers (RTC_TR and RTC_DR) are synchronized with RTC APB clock.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that

contains the configuration information for RTC.

- | | |
|---------------|---|
| Return values | <ul style="list-style-type: none"> • HAL: status |
| Notes | <ul style="list-style-type: none"> • The RTC Resynchronization mode is write protected, use the <code>__HAL_RTC_WRITEPROTECTION_DISABLE()</code> before calling this function. • To read the calendar through the shadow registers after Calendar initialization, calendar update or after wakeup from low power modes the software must first clear the RSF flag. The software must then wait until it is set again before reading the calendar, which means that the calendar registers have been correctly copied into the RTC_TR and RTC_DR shadow registers. |

HAL_RTC_GetState

- | | |
|----------------------|--|
| Function name | HAL_RTCStateTypeDef HAL_RTC_GetState
(RTC_HandleTypeDef * hrtc) |
| Function description | Returns the RTC state. |
| Parameters | <ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. |
| Return values | <ul style="list-style-type: none"> • HAL: state |

RTC_EnterInitMode

- | | |
|----------------------|--|
| Function name | HAL_StatusTypeDef RTC_EnterInitMode
(RTC_HandleTypeDef * hrtc) |
| Function description | Enters the RTC Initialization mode. |
| Parameters | <ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. |
| Return values | <ul style="list-style-type: none"> • HAL: status |
| Notes | <ul style="list-style-type: none"> • The RTC Initialization mode is write protected, use the <code>__HAL_RTC_WRITEPROTECTION_DISABLE()</code> before calling this function. |

RTC_ByteToBcd2

- | | |
|----------------------|--|
| Function name | uint8_t RTC_ByteToBcd2 (uint8_t Value) |
| Function description | Converts a 2 digit decimal to BCD format. |
| Parameters | <ul style="list-style-type: none"> • Value: Byte to be converted |
| Return values | <ul style="list-style-type: none"> • Converted: byte |

RTC_Bcd2ToByte

- | | |
|----------------------|---|
| Function name | uint8_t RTC_Bcd2ToByte (uint8_t Value) |
| Function description | Converts from 2 digit BCD to Binary. |
| Parameters | <ul style="list-style-type: none"> • Value: BCD value to be converted |

Return values • **Converted:** word

36.3 RTC Firmware driver defines

36.3.1 RTC

AlarmDateWeekDay Definitions

RTC_ALARMDATEWEEKDAYSEL_DATE
RTC_ALARMDATEWEEKDAYSEL_WEEKDAY
IS_RTC_ALARM_DATE_WEEKDAY_SEL

Alarm Mask Definitions

RTC_ALARMMASK_NONE
RTC_ALARMMASK_DATEWEEKDAY
RTC_ALARMMASK_HOURS
RTC_ALARMMASK_MINUTES
RTC_ALARMMASK_SECONDS
RTC_ALARMMASK_ALL
IS_RTC_ALARM_MASK

Alarms Definitions

RTC_ALARM_A
RTC_ALARM_B
IS_RTC_ALARM

Alarm Definitions

IS_RTC_ALARM_DATE_WEEKDAY_DATE
IS_RTC_ALARM_DATE_WEEKDAY_WEEKDAY

Alarm Sub Seconds Masks Definitions

RTC_ALARMSUBSECONDMASK_ALL	All Alarm SS fields are masked. There is no comparison on sub seconds for Alarm
RTC_ALARMSUBSECONDMASK_SS14_1	SS[14:1] are don't care in Alarm comparison. Only SS[0] is compared.
RTC_ALARMSUBSECONDMASK_SS14_2	SS[14:2] are don't care in Alarm comparison. Only SS[1:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_3	SS[14:3] are don't care in Alarm comparison. Only SS[2:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_4	SS[14:4] are don't care in Alarm comparison. Only SS[3:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_5	SS[14:5] are don't care in Alarm comparison. Only SS[4:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_6	SS[14:6] are don't care in Alarm comparison. Only SS[5:0] are compared

RTC_ALARMSUBSECONDMASK_SS14_7	SS[14:7] are don't care in Alarm comparison. Only SS[6:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_8	SS[14:8] are don't care in Alarm comparison. Only SS[7:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_9	SS[14:9] are don't care in Alarm comparison. Only SS[8:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_10	SS[14:10] are don't care in Alarm comparison. Only SS[9:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_11	SS[14:11] are don't care in Alarm comparison. Only SS[10:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_12	SS[14:12] are don't care in Alarm comparison. Only SS[11:0] are compared
RTC_ALARMSUBSECONDMASK_SS14_13	SS[14:13] are don't care in Alarm comparison. Only SS[12:0] are compared
RTC_ALARMSUBSECONDMASK_SS14	SS[14] is don't care in Alarm comparison. Only SS[13:0] are compared
RTC_ALARMSUBSECONDMASK_NONE	SS[14:0] are compared and must match to activate alarm.

IS_RTC_ALARM_SUB_SECOND_MASK

Alarm Sub Seconds Value

IS_RTC_ALARM_SUB_SECOND_VALUE

AM PM Definitions

RTC_HOURFORMAT12_AM

RTC_HOURFORMAT12_PM

IS_RTC_HOURFORMAT12

Asynchronous Predivider

IS_RTC_ASYNCH_PREDIV

DayLightSaving

RTC_DAYLIGHTSAVING_SUB1H

RTC_DAYLIGHTSAVING_ADD1H

RTC_DAYLIGHTSAVING_NONE

IS_RTC_DAYLIGHT_SAVING

RTC Exported Macros

_HAL_RTC_RESET_HANDLE_STATE

Description:

- Reset RTC handle state.

Parameters:

- **_HANDLE_**: RTC handle.

Return value:

- None

`__HAL_RTC_WRITEPROTECTION_DISABLE`

Description:

- Disable the write protection for RTC registers.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`__HAL_RTC_WRITEPROTECTION_ENABLE`

Description:

- Enable the write protection for RTC registers.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`__HAL_RTC_ALARMA_ENABLE`

Description:

- Enable the RTC ALARMA peripheral.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`__HAL_RTC_ALARMA_DISABLE`

Description:

- Disable the RTC ALARMA peripheral.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`__HAL_RTC_ALARMB_ENABLE`

Description:

- Enable the RTC ALARMB peripheral.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`_HAL_RTC_ALARMB_DISABLE`

Description:

- Disable the RTC ALARMB peripheral.

Parameters:

- `_HANDLE_`: specifies the RTC handle.

Return value:

- None

`_HAL_RTC_ALARM_ENABLE_IT`

Description:

- Enable the RTC Alarm interrupt.

Parameters:

- `_HANDLE_`: specifies the RTC handle.
- `_INTERRUPT_`: specifies the RTC Alarm interrupt sources to be enabled or disabled. This parameter can be any combination of the following values:
 - `RTC_IT_ALRA`: Alarm A interrupt
 - `RTC_IT_ALRB`: Alarm B interrupt

Return value:

- None

`_HAL_RTC_ALARM_DISABLE_IT`

Description:

- Disable the RTC Alarm interrupt.

Parameters:

- `_HANDLE_`: specifies the RTC handle.
- `_INTERRUPT_`: specifies the RTC Alarm interrupt sources to be enabled or disabled. This parameter can be any combination of the following values:
 - `RTC_IT_ALRA`: Alarm A interrupt
 - `RTC_IT_ALRB`: Alarm B interrupt

Return value:

- None

[__HAL_RTC_ALARM_GET_IT](#)**Description:**

- Check whether the specified RTC Alarm interrupt has occurred or not.

Parameters:

- [__HANDLE__](#): specifies the RTC handle.
- [__INTERRUPT__](#): specifies the RTC Alarm interrupt sources to check. This parameter can be:
 - [RTC_IT_ALRA](#): Alarm A interrupt
 - [RTC_IT_ALRB](#): Alarm B interrupt

Return value:

- None

[__HAL_RTC_ALARM_GET_IT_SOURCE](#)**Description:**

- Check whether the specified RTC Alarm interrupt has been enabled or not.

Parameters:

- [__HANDLE__](#): specifies the RTC handle.
- [__INTERRUPT__](#): specifies the RTC Alarm interrupt sources to check. This parameter can be:
 - [RTC_IT_ALRA](#): Alarm A interrupt
 - [RTC_IT_ALRB](#): Alarm B interrupt

Return value:

- None

[__HAL_RTC_ALARM_GET_FLAG](#)**Description:**

- Get the selected RTC Alarm's flag status.

Parameters:

- [__HANDLE__](#): specifies the RTC handle.
- [__FLAG__](#): specifies the RTC Alarm Flag sources to check. This parameter can be:
 - [RTC_FLAG_ALRAF](#)
 - [RTC_FLAG_ALRBF](#)
 - [RTC_FLAG_ALRAWF](#)
 - [RTC_FLAG_ALRBWF](#)

Return value:

- None

`__HAL_RTC_ALARM_CLEAR_FLAG`

Description:

- Clear the RTC Alarm's pending flags.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__FLAG__`: specifies the RTC Alarm Flag sources to clear. This parameter can be:
 - `RTC_FLAG_ALRAF`
 - `RTC_FLAG_ALRBF`

Return value:

- None

`__HAL_RTC_ALARM_EXTI_ENABLE_IT`

Description:

- Enable interrupt on the RTC Alarm associated Exti line.

Return value:

- None

`__HAL_RTC_ALARM_EXTI_DISABLE_IT`

Description:

- Disable interrupt on the RTC Alarm associated Exti line.

Return value:

- None

`__HAL_RTC_ALARM_EXTI_ENABLE_EVENT`

Description:

- Enable event on the RTC Alarm associated Exti line.

Return value:

- None.

`__HAL_RTC_ALARM_EXTI_DISABLE_EVENT`

Description:

- Disable event on the RTC Alarm associated Exti line.

Return value:

- None.

`__HAL_RTC_ALARM_EXTI_ENABLE_FALLING_EDGE`

Description:

- Enable falling edge trigger on the RTC Alarm associated Exti line.

Return value:

- None.

Description:

- Disable falling edge trigger on the RTC Alarm associated Exti line.

Return value:

- None.

Description:

- Enable rising edge trigger on the RTC Alarm associated Exti line.

Return value:

- None.

Description:

- Disable rising edge trigger on the RTC Alarm associated Exti line.

Return value:

- None.

Description:

- Enable rising & falling edge trigger on the RTC Alarm associated Exti line.

Return value:

- None.

Description:

- Disable rising & falling edge trigger on the RTC Alarm associated Exti line.

Return value:

- None.

Description:

- Check whether the RTC Alarm associated Exti line interrupt flag is set or not.

Return value:

- Line: Status.

Description:

- Clear the RTC Alarm

associated Exti line flag.

Return value:

- None.

Description:

- Generate a Software interrupt on RTC Alarm associated Exti line.

Return value:

- None.

Flags Definitions

RTC_FLAG_RECALPF
RTC_FLAG_TAMP3F
RTC_FLAG_TAMP2F
RTC_FLAG_TAMP1F
RTC_FLAG_TSOVF
RTC_FLAG_TSF
RTC_FLAG_WUTF
RTC_FLAG_ALRBF
RTC_FLAG_ALRAF
RTC_FLAG_INITF
RTC_FLAG_RSF
RTC_FLAG_INITS
RTC_FLAG_SHPF
RTC_FLAG_WUTWF
RTC_FLAG_ALRBWF
RTC_FLAG_ALRAWF

Hour Formats

RTC_HOURFORMAT_24
RTC_HOURFORMAT_12
IS_RTC_HOUR_FORMAT

Input Parameter Format

RTC_FORMAT_BIN
RTC_FORMAT_BCD
IS_RTC_FORMAT

Interrupts Definitions

RTC_IT_TS
RTC_IT_WUT

RTC_IT_ALRB
RTC_IT_ALRA
RTC_IT_TAMP1
RTC_IT_TAMP2
RTC_IT_TAMP3

Masks Definitions

RTC_TR_RESERVED_MASK
RTC_DR_RESERVED_MASK
RTC_INIT_MASK
RTC_RSF_MASK
RTC_FLAGS_MASK

Month Definitions

RTC_MONTH_JANUARY
RTC_MONTH_FEBRUARY
RTC_MONTH_MARCH
RTC_MONTH_APRIIL
RTC_MONTH_MAY
RTC_MONTH_JUNE
RTC_MONTH_JULY
RTC_MONTH_AUGUST
RTC_MONTH_SEPTMBER
RTC_MONTH_OCTOBER
RTC_MONTH_NOVEMBER
RTC_MONTH_DECEMBER

IS_RTC_MONTH

IS_RTC_DATE

Output Polarity

RTC_OUTPUT_POLARITY_HIGH
RTC_OUTPUT_POLARITY_LOW
IS_RTC_OUTPUT_POL

Alarm Output Type

RTC_OUTPUT_TYPE_OPENDRAIN
RTC_OUTPUT_TYPE_PUSH_PULL
IS_RTC_OUTPUT_TYPE

StoreOperation

RTC_STOREOPERATION_RESET

RTC_STOREOPERATION_SET
IS_RTC_STORE_OPERATION
Synchronous Predivider
IS_RTC_SYNCH_PREDIV
Default Timeout Value
RTC_TIMEOUT_VALUE
Time Definitions
IS_RTC_HOUR12
IS_RTC_HOUR24
IS_RTC_MINUTES
IS_RTC_SECONDS
WeekDay Definitions
RTC_WEEKDAY_MONDAY
RTC_WEEKDAY_TUESDAY
RTC_WEEKDAY_WEDNESDAY
RTC_WEEKDAY_THURSDAY
RTC_WEEKDAY_FRIDAY
RTC_WEEKDAY_SATURDAY
RTC_WEEKDAY_SUNDAY
IS_RTC_WEEKDAY
Year Definitions
IS_RTC_YEAR

37 HAL RTC Extension Driver

37.1 RTCEx Firmware driver registers structures

37.1.1 RTC_TamperTypeDef

Data Fields

- *uint32_t Tamper*
- *uint32_t Trigger*
- *uint32_t Filter*
- *uint32_t SamplingFrequency*
- *uint32_t PrechargeDuration*
- *uint32_t TamperPullUp*
- *uint32_t TimeStampOnTamperDetection*

Field Documentation

- *uint32_t RTC_TamperTypeDef::Tamper*
Specifies the Tamper Pin. This parameter can be a value of
[*RTCEx_Tamper_Pins_Definitions*](#)
- *uint32_t RTC_TamperTypeDef::Trigger*
Specifies the Tamper Trigger. This parameter can be a value of
[*RTCEx_Tamper_Trigger_Definitions*](#)
- *uint32_t RTC_TamperTypeDef::Filter*
Specifies the RTC Filter Tamper. This parameter can be a value of
[*RTCEx_Tamper_Filter_Definitions*](#)
- *uint32_t RTC_TamperTypeDef::SamplingFrequency*
Specifies the sampling frequency. This parameter can be a value of
[*RTCEx_Tamper_Sampling_Frequencies_Definitions*](#)
- *uint32_t RTC_TamperTypeDef::PrechargeDuration*
Specifies the Precharge Duration . This parameter can be a value of
[*RTCEx_Tamper_Pin_Precharge_Duration_Definitions*](#)
- *uint32_t RTC_TamperTypeDef::TamperPullUp*
Specifies the Tamper PullUp . This parameter can be a value of
[*RTCEx_Tamper_Pull_Up_Definitions*](#)
- *uint32_t RTC_TamperTypeDef::TimeStampOnTamperDetection*
Specifies the TimeStampOnTamperDetection. This parameter can be a value of
[*RTCEx_Tamper_TimeStampOnTamperDetection_Definitions*](#)

37.1.2 RTC_TimeTypeDef

Data Fields

- *uint8_t Hours*
- *uint8_t Minutes*
- *uint8_t Seconds*
- *uint8_t TimeFormat*
- *uint32_t SubSeconds*
- *uint32_t SecondFraction*
- *uint32_t DayLightSaving*
- *uint32_t StoreOperation*

Field Documentation

- ***uint8_t RTC_TimeTypeDef::Hours***
Specifies the RTC Time Hour. This parameter must be a number between Min_Data = 0 and Max_Data = 12 if the RTC_HourFormat_12 is selected. This parameter must be a number between Min_Data = 0 and Max_Data = 23 if the RTC_HourFormat_24 is selected
- ***uint8_t RTC_TimeTypeDef::Minutes***
Specifies the RTC Time Minutes. This parameter must be a number between Min_Data = 0 and Max_Data = 59
- ***uint8_t RTC_TimeTypeDef::Seconds***
Specifies the RTC Time Seconds. This parameter must be a number between Min_Data = 0 and Max_Data = 59
- ***uint8_t RTC_TimeTypeDef::TimeFormat***
Specifies the RTC AM/PM Time. This parameter can be a value of [**RTC_AM_PM_Definitions**](#)
- ***uint32_t RTC_TimeTypeDef::SubSeconds***
Specifies the RTC_SSR RTC Sub Second register content. This parameter corresponds to a time unit range between [0-1] Second with [1 Sec / SecondFraction +1] granularity
- ***uint32_t RTC_TimeTypeDef::SecondFraction***
Specifies the range or granularity of Sub Second register content corresponding to Synchronous pre-scaler factor value (PREDIV_S) This parameter corresponds to a time unit range between [0-1] Second with [1 Sec / SecondFraction +1] granularity. This field will be used only by HAL_RTC_GetTime function
- ***uint32_t RTC_TimeTypeDef::DayLightSaving***
Specifies RTC_DayLightSaveOperation: the value of hour adjustment. This parameter can be a value of [**RTC_DayLightSaving_Definitions**](#)
- ***uint32_t RTC_TimeTypeDef::StoreOperation***
Specifies RTC_StoreOperation value to be written in the BKP bit in CR register to store the operation. This parameter can be a value of [**RTC_StoreOperation_Definitions**](#)

37.1.3 RTC_AlarmTypeDef

Data Fields

- ***RTC_TimeTypeDef AlarmTime***
- ***uint32_t AlarmMask***
- ***uint32_t AlarmSubSecondMask***
- ***uint32_t AlarmDateWeekDaySel***
- ***uint8_t AlarmDateWeekDay***
- ***uint32_t Alarm***

Field Documentation

- ***RTC_TimeTypeDef RTC_AlarmTypeDef::AlarmTime***
Specifies the RTC Alarm Time members
- ***uint32_t RTC_AlarmTypeDef::AlarmMask***
Specifies the RTC Alarm Masks. This parameter can be a value of [**RTC_AlarmMask_Definitions**](#)
- ***uint32_t RTC_AlarmTypeDef::AlarmSubSecondMask***
Specifies the RTC Alarm SubSeconds Masks. This parameter can be a value of [**RTC_Alarm_Sub_Seconds_Masks_Definitions**](#)
- ***uint32_t RTC_AlarmTypeDef::AlarmDateWeekDaySel***
Specifies the RTC Alarm is on Date or WeekDay. This parameter can be a value of [**RTC_AlarmDateWeekDay_Definitions**](#)

- **`uint8_t RTC_AlarmTypeDef::AlarmDateWeekDay`**
Specifies the RTC Alarm Date/WeekDay. If the Alarm Date is selected, this parameter must be set to a value in the 1-31 range. If the Alarm WeekDay is selected, this parameter can be a value of [RTC_WeekDay_Definitions](#)
- **`uint32_t RTC_AlarmTypeDef::Alarm`**
Specifies the alarm . This parameter can be a value of [RTC_Alarms_Definitions](#)

37.2 RTCEEx Firmware driver API description

37.2.1 How to use this driver

- Enable the RTC domain access.
- Configure the RTC Prescaler (Asynchronous and Synchronous) and RTC hour format using the HAL_RTC_Init() function.

RTC Wakeup configuration

- To configure the RTC Wakeup Clock source and Counter use the HAL_RTCEEx_SetWakeUpTimer() function. You can also configure the RTC Wakeup timer with interrupt mode using the HAL_RTCEEx_SetWakeUpTimer_IT() function.
- To read the RTC WakeUp Counter register, use the HAL_RTCEEx_GetWakeUpTimer() function.

TimeStamp configuration

- Configure the RTC_AFx trigger and enable the RTC TimeStamp using the HAL_RTCEEx_SetTimeStamp() function. You can also configure the RTC TimeStamp with interrupt mode using the HAL_RTCEEx_SetTimeStamp_IT() function.
- To read the RTC TimeStamp Time and Date register, use the HAL_RTCEEx_GetTimeStamp() function.
- The TIMESTAMP alternate function can be mapped to RTC_AF1 (PC13).

Tamper configuration

- Enable the RTC Tamper and configure the Tamper filter count, trigger Edge or Level according to the Tamper filter (if equal to 0 Edge else Level) value, sampling frequency, precharge or discharge and Pull-UP using the HAL_RTCEEx_SetTamper() function. You can configure RTC Tamper with interrupt mode using HAL_RTCEEx_SetTamper_IT() function.
- The TAMPER1 alternate function can be mapped to RTC_AF1 (PC13).

Backup Data Registers configuration

- To write to the RTC Backup Data registers, use the HAL_RTCEEx_BKUPWrite() function.
- To read the RTC Backup Data registers, use the HAL_RTCEEx_BKUPRead() function.

37.2.2 RTC TimeStamp and Tamper functions

This section provides functions allowing to configure TimeStamp feature

This section contains the following APIs:

- [`HAL_RTCEEx_SetTimeStamp\(\)`](#)
- [`HAL_RTCEEx_SetTimeStamp_IT\(\)`](#)
- [`HAL_RTCEEx_DeactivateTimeStamp\(\)`](#)
- [`HAL_RTCEEx_GetTimeStamp\(\)`](#)

- `HAL_RTCEEx_SetTamper()`
- `HAL_RTCEEx_SetTamper_IT()`
- `HAL_RTCEEx_DeactivateTamper()`
- `HAL_RTCEEx_TamperTimeStampIRQHandler()`
- `HAL_RTCEEx_TimeStampEventCallback()`
- `HAL_RTCEEx_Tamper1EventCallback()`
- `HAL_RTCEEx_Tamper2EventCallback()`
- `HAL_RTCEEx_Tamper3EventCallback()`
- `HAL_RTCEEx_PollForTimeStampEvent()`
- `HAL_RTCEEx_PollForTamper1Event()`
- `HAL_RTCEEx_PollForTamper2Event()`
- `HAL_RTCEEx_PollForTamper3Event()`

37.2.3 RTC Wake-up functions

This section provides functions allowing to configure Wake-up feature

This section contains the following APIs:

- `HAL_RTCEEx_SetWakeUpTimer()`
- `HAL_RTCEEx_SetWakeUpTimer_IT()`
- `HAL_RTCEEx_DeactivateWakeUpTimer()`
- `HAL_RTCEEx_GetWakeUpTimer()`
- `HAL_RTCEEx_WakeUpTimerIRQHandler()`
- `HAL_RTCEEx_WakeUpTimerEventCallback()`
- `HAL_RTCEEx_PollForWakeUpTimerEvent()`

37.2.4 Extension Peripheral Control functions

This subsection provides functions allowing to

- Writes a data in a specified RTC Backup data register
- Read a data in a specified RTC Backup data register
- Sets the Coarse calibration parameters.
- Deactivates the Coarse calibration parameters
- Sets the Smooth calibration parameters.
- Configures the Synchronization Shift Control Settings.
- Configures the Calibration Pinout (RTC_CALIB) Selection (1Hz or 512Hz).
- Deactivates the Calibration Pinout (RTC_CALIB) Selection (1Hz or 512Hz).
- Enables the RTC reference clock detection.
- Disable the RTC reference clock detection.
- Enables the Bypass Shadow feature.
- Disables the Bypass Shadow feature.

This section contains the following APIs:

- `HAL_RTCEEx_BKUPWrite()`
- `HAL_RTCEEx_BKUPRead()`
- `HAL_RTCEEx_SetCoarseCalib()`
- `HAL_RTCEEx_DeactivateCoarseCalib()`
- `HAL_RTCEEx_SetSmoothCalib()`
- `HAL_RTCEEx_SetSynchroShift()`
- `HAL_RTCEEx_SetCalibrationOutPut()`
- `HAL_RTCEEx_DeactivateCalibrationOutPut()`
- `HAL_RTCEEx_SetRefClock()`
- `HAL_RTCEEx_DeactivateRefClock()`

- [*HAL_RTCEx_EnableBypassShadow\(\)*](#)
- [*HAL_RTCEx_DisableBypassShadow\(\)*](#)

37.2.5 Extended features functions

This section provides functions allowing to:

- RTC Alram B callback
- RTC Poll for Alarm B request

This section contains the following APIs:

- [*HAL_RTCEx_AlarmBEventCallback\(\)*](#)
- [*HAL_RTCEx_PollForAlarmBEvent\(\)*](#)

37.2.6 Detailed description of functions

HAL_RTCEx_SetTimeStamp

Function name **HAL_StatusTypeDef HAL_RTCEx_SetTimeStamp
(RTC_HandleTypeDef * hrtc, uint32_t TimeStampEdge)**

Function description SetsTimeStamp.

- Parameters
- **hrtc:** pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
 - **TimeStampEdge:** Specifies the pin edge on which the TimeStamp is activated. This parameter can be one of the following values:
 - RTC_TIMESTAMPEDGE_RISING: the Time stamp event occurs on the rising edge of the related pin.
 - RTC_TIMESTAMPEDGE_FALLING: the Time stamp event occurs on the falling edge of the related pin.

- Return values
- **HAL:** status

- Notes
- This API must be called before enabling the TimeStamp feature.

HAL_RTCEx_SetTimeStamp_IT

Function name **HAL_StatusTypeDef HAL_RTCEx_SetTimeStamp_IT
(RTC_HandleTypeDef * hrtc, uint32_t TimeStampEdge)**

Function description SetsTimeStamp with Interrupt.

- Parameters
- **hrtc:** pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
 - **TimeStampEdge:** Specifies the pin edge on which the TimeStamp is activated. This parameter can be one of the following values:
 - RTC_TIMESTAMPEDGE_RISING: the Time stamp event occurs on the rising edge of the related pin.
 - RTC_TIMESTAMPEDGE_FALLING: the Time stamp event occurs on the falling edge of the related pin.

- Return values
- **HAL:** status

- Notes
- This API must be called before enabling the TimeStamp feature.

HAL_RTCEx_DeactivateTimeStamp

Function name	HAL_StatusTypeDef HAL_RTCEx_DeactivateTimeStamp (RTC_HandleTypeDef * hrtc)
Function description	Deactivates TimeStamp.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_GetTimeStamp

Function name	HAL_StatusTypeDef HAL_RTCEx_GetTimeStamp (RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTimeStamp, RTC_DateTypeDef * sTimeStampDate, uint32_t Format)
Function description	Gets the RTC TimeStamp value.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • sTimeStamp: Pointer to Time structure • sTimeStampDate: Pointer to Date structure • Format: specifies the format of the entered parameters. This parameter can be one of the following values: RTC_FORMAT_BIN: Binary data format RTC_FORMAT_BCD: BCD data format
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_SetTamper

Function name	HAL_StatusTypeDef HAL_RTCEx_SetTamper (RTC_HandleTypeDef * hrtc, RTC_TamperTypeDef * sTamper)
Function description	Sets Tamper.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • sTamper: Pointer to Tamper Structure.
Return values	<ul style="list-style-type: none"> • HAL: status

Notes

- By calling this API we disable the tamper interrupt for all tampers.

HAL_RTCEx_SetTamper_IT

Function name	HAL_StatusTypeDef HAL_RTCEx_SetTamper_IT (RTC_HandleTypeDef * hrtc, RTC_TamperTypeDef * sTamper)
Function description	Sets Tamper with interrupt.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • sTamper: Pointer to RTC Tamper.
Return values	<ul style="list-style-type: none"> • HAL: status

- Notes
- By calling this API we force the tamper interrupt for all tampers.

HAL_RTCEx_DeactivateTamper

Function name	HAL_StatusTypeDef HAL_RTCEx_DeactivateTamper (RTC_HandleTypeDef * hrtc, uint32_t Tamper)
Function description	Deactivates Tamper.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • Tamper: Selected tamper pin. This parameter can be a value of Tamper Pins Definitions
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_TamperTimeStampIRQHandler

Function name	void HAL_RTCEx_TamperTimeStampIRQHandler (RTC_HandleTypeDef * hrtc)
Function description	This function handles TimeStamp interrupt request.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTCEx_Tamper1EventCallback

Function name	void HAL_RTCEx_Tamper1EventCallback (RTC_HandleTypeDef * hrtc)
Function description	Tamper 1 callback.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTCEx_Tamper2EventCallback

Function name	void HAL_RTCEx_Tamper2EventCallback (RTC_HandleTypeDef * hrtc)
Function description	Tamper 2 callback.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTCEx_Tamper3EventCallback

Function name	void HAL_RTCEx_Tamper3EventCallback (RTC_HandleTypeDef * hrtc)
Function description	Tamper 3 callback.

Parameters	<ul style="list-style-type: none">• hrtc: RTC handle
Return values	<ul style="list-style-type: none">• None:

HAL_RTCEx_TimeStampEventCallback

Function name	void HAL_RTCEx_TimeStampEventCallback (RTC_HandleTypeDef * hrtc)
Function description	TimeStamp callback.
Parameters	<ul style="list-style-type: none">• hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none">• None:

HAL_RTCEx_PollForTimeStampEvent

Function name	HAL_StatusTypeDef HAL_RTCEx_PollForTimeStampEvent (RTC_HandleTypeDef * hrtc, uint32_t Timeout)
Function description	This function handles TimeStamp polling request.
Parameters	<ul style="list-style-type: none">• hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.• Timeout: Timeout duration
Return values	<ul style="list-style-type: none">• HAL: status

HAL_RTCEx_PollForTamper1Event

Function name	HAL_StatusTypeDef HAL_RTCEx_PollForTamper1Event (RTC_HandleTypeDef * hrtc, uint32_t Timeout)
Function description	This function handles Tamper1 Polling.
Parameters	<ul style="list-style-type: none">• hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.• Timeout: Timeout duration
Return values	<ul style="list-style-type: none">• HAL: status

HAL_RTCEx_PollForTamper2Event

Function name	HAL_StatusTypeDef HAL_RTCEx_PollForTamper2Event (RTC_HandleTypeDef * hrtc, uint32_t Timeout)
Function description	This function handles Tamper2 Polling.
Parameters	<ul style="list-style-type: none">• hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.• Timeout: Timeout duration
Return values	<ul style="list-style-type: none">• HAL: status

HAL_RTCEx_PollForTamper3Event

Function name	HAL_StatusTypeDef HAL_RTCEx_PollForTamper3Event (RTC_HandleTypeDef * hrtc, uint32_t Timeout)
---------------	---

Function description	This function handles Tamper3 Polling.
Parameters	<ul style="list-style-type: none"> • hrtc: RTC handle • Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_SetWakeUpTimer

Function name	HAL_StatusTypeDef HAL_RTCEx_SetWakeUpTimer (RTC_HandleTypeDef * hrtc, uint32_t WakeUpCounter, uint32_t WakeUpClock)
Function description	Set wake up timer.
Parameters	<ul style="list-style-type: none"> • hrtc: RTC handle • WakeUpCounter: Wake up counter • WakeUpClock: Wake up clock
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_SetWakeUpTimer_IT

Function name	HAL_StatusTypeDef HAL_RTCEx_SetWakeUpTimer_IT (RTC_HandleTypeDef * hrtc, uint32_t WakeUpCounter, uint32_t WakeUpClock)
Function description	Set wake up timer with interrupt.
Parameters	<ul style="list-style-type: none"> • hrtc: RTC handle • WakeUpCounter: Wake up counter • WakeUpClock: Wake up clock
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_DeactivateWakeUpTimer

Function name	uint32_t HAL_RTCEx_DeactivateWakeUpTimer (RTC_HandleTypeDef * hrtc)
Function description	Deactivates wake up timer counter.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.

Return values	<ul style="list-style-type: none"> • HAL: status
---------------	--

HAL_RTCEx_GetWakeUpTimer

Function name	uint32_t HAL_RTCEx_GetWakeUpTimer (RTC_HandleTypeDef * hrtc)
Function description	Gets wake up timer counter.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.

Return values	<ul style="list-style-type: none"> • Counter: value
---------------	---

HAL_RTCEx_WakeUpTimerIRQHandler

Function name	void HAL_RTCEx_WakeUpTimerIRQHandler (RTC_HandleTypeDef * hrtc)
Function description	This function handles Wake Up Timer interrupt request.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTCEx_WakeUpTimerEventCallback

Function name	void HAL_RTCEx_WakeUpTimerEventCallback (RTC_HandleTypeDef * hrtc)
Function description	Wake Up Timer callback.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTCEx_PollForWakeUpTimerEvent

Function name	HAL_StatusTypeDef HAL_RTCEx_PollForWakeUpTimerEvent (RTC_HandleTypeDef * hrtc, uint32_t Timeout)
Function description	This function handles Wake Up Timer Polling.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_BKUPWrite

Function name	void HAL_RTCEx_BKUPWrite (RTC_HandleTypeDef * hrtc, uint32_t BackupRegister, uint32_t Data)
Function description	Writes a data in a specified RTC Backup data register.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • BackupRegister: RTC Backup data Register number. This parameter can be: RTC_BKP_DRx where x can be from 0 to 19 to specify the register. • Data: Data to be written in the specified RTC Backup data register.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTCEx_BKUPRead

Function name	uint32_t HAL_RTCEx_BKUPRead (RTC_HandleTypeDef * hrtc, uint32_t BackupRegister)
Function description	Reads data from the specified RTC Backup data Register.

Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. BackupRegister: RTC Backup data Register number. This parameter can be: RTC_BKP_DRx where x can be from 0 to 19 to specify the register.
Return values	<ul style="list-style-type: none"> Read: value

HAL_RTCEx_SetCoarseCalib

Function name	HAL_StatusTypeDef HAL_RTCEx_SetCoarseCalib (RTC_HandleTypeDef * hrtc, uint32_t CalibSign, uint32_t Value)
Function description	Sets the Coarse calibration parameters.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. CalibSign: Specifies the sign of the coarse calibration value. This parameter can be one of the following values : <ul style="list-style-type: none"> RTC_CALIBSIGN_POSITIVE: The value sign is positive RTC_CALIBSIGN_NEGATIVE: The value sign is negative Value: value of coarse calibration expressed in ppm (coded on 5 bits).
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> This Calibration value should be between 0 and 63 when using negative sign with a 2-ppm step. This Calibration value should be between 0 and 126 when using positive sign with a 4-ppm step.

HAL_RTCEx_DeactivateCoarseCalib

Function name	HAL_StatusTypeDef HAL_RTCEx_DeactivateCoarseCalib (RTC_HandleTypeDef * hrtc)
Function description	Deactivates the Coarse calibration parameters.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> HAL: status

HAL_RTCEx_SetSmoothCalib

Function name	HAL_StatusTypeDef HAL_RTCEx_SetSmoothCalib (RTC_HandleTypeDef * hrtc, uint32_t SmoothCalibPeriod, uint32_t SmoothCalibPlusPulses, uint32_t SmoothCalibMinusPulsesValue)
Function description	Sets the Smooth calibration parameters.
Parameters	<ul style="list-style-type: none"> hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. SmoothCalibPeriod: Select the Smooth Calibration Period. This parameter can be one of the following values : <ul style="list-style-type: none"> RTC_SMOOTHCALIB_PERIOD_32SEC: The smooth

	<ul style="list-style-type: none"> - calibration periode is 32s. - RTC_SMOOTHCALIB_PERIOD_16SEC: The smooth calibration periode is 16s. - RTC_SMOOTHCALIB_PERIOD_8SEC: The smooth calibartion periode is 8s.
	<ul style="list-style-type: none"> • SmoothCalibPlusPulses: Select to Set or reset the CALP bit. This parameter can be one of the following values: <ul style="list-style-type: none"> - RTC_SMOOTHCALIB_PLUSPULSES_SET: Add one RTCCLK puls every 2^{11} pulses. - RTC_SMOOTHCALIB_PLUSPULSES_RESET: No RTCCLK pulses are added. • SmoothCalibMinusPulsesValue: Select the value of CALM[8:0] bits. This parameter can be one any value from 0 to 0x000001FF.
Return values	• HAL: status

Notes

- To deactivate the smooth calibration, the field SmoothCalibPlusPulses must be equal to SMOOTHCALIB_PLUSPULSES_RESET and the field SmoothCalibMinusPulsesValue mut be equal to 0.

HAL_RTCEx_SetSynchroShift

Function name	HAL_StatusTypeDef HAL_RTCEx_SetSynchroShift (RTC_HandleTypeDef * hrtc, uint32_t ShiftAdd1S, uint32_t ShiftSubFS)
Function description	Configures the Synchronization Shift Control Settings.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • ShiftAdd1S: Select to add or not 1 second to the time calendar. This parameter can be one of the following values : <ul style="list-style-type: none"> - RTC_SHIFTADD1S_SET: Add one second to the clock calendar. - RTC_SHIFTADD1S_RESET: No effect. • ShiftSubFS: Select the number of Second Fractions to substitute. This parameter can be one any value from 0 to 0x7FFF.
Return values	• HAL: status
Notes	<ul style="list-style-type: none"> • When REFCKON is set, firmware must not write to Shift control register.

HAL_RTCEx_SetCalibrationOutPut

Function name	HAL_StatusTypeDef HAL_RTCEx_SetCalibrationOutPut (RTC_HandleTypeDef * hrtc, uint32_t CalibOutput)
Function description	Configures the Calibration Pinout (RTC_CALIB) Selection (1Hz or 512Hz).
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • CalibOutput: : Select the Calibration output Selection . This parameter can be one of the following values:

- RTC_CALIBOUTPUT_512HZ: A signal has a regular waveform at 512Hz.
- RTC_CALIBOUTPUT_1HZ: A signal has a regular waveform at 1Hz.

Return values

- **HAL:** status

HAL_RTCEx_DeactivateCalibrationOutPut

Function name	HAL_StatusTypeDef HAL_RTCEx_DeactivateCalibrationOutPut (RTC_HandleTypeDef * hrtc)
Function description	Deactivates the Calibration Pinout (RTC_CALIB) Selection (1Hz or 512Hz).
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_SetRefClock

Function name	HAL_StatusTypeDef HAL_RTCEx_SetRefClock (RTC_HandleTypeDef * hrtc)
Function description	Enables the RTC reference clock detection.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_DeactivateRefClock

Function name	HAL_StatusTypeDef HAL_RTCEx_DeactivateRefClock (RTC_HandleTypeDef * hrtc)
Function description	Disable the RTC reference clock detection.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_RTCEx_EnableBypassShadow

Function name	HAL_StatusTypeDef HAL_RTCEx_EnableBypassShadow (RTC_HandleTypeDef * hrtc)
Function description	Enables the Bypass Shadow feature.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When the Bypass Shadow is enabled the calendar value are taken directly from the Calendar counter.

HAL_RTCEx_DisableBypassShadow

Function name	HAL_StatusTypeDef HAL_RTCEx_DisableBypassShadow (RTC_HandleTypeDef * hrtc)
Function description	Disables the Bypass Shadow feature.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When the Bypass Shadow is enabled the calendar value are taken directly from the Calendar counter.

HAL_RTCEx_AlarmBEventCallback

Function name	void HAL_RTCEx_AlarmBEventCallback (RTC_HandleTypeDef * hrtc)
Function description	Alarm B callback.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.
Return values	<ul style="list-style-type: none"> • None:

HAL_RTCEx_PollForAlarmBEvent

Function name	HAL_StatusTypeDef HAL_RTCEx_PollForAlarmBEvent (RTC_HandleTypeDef * hrtc, uint32_t Timeout)
Function description	This function handles AlarmB Polling request.
Parameters	<ul style="list-style-type: none"> • hrtc: pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC. • Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> • HAL: status

37.3 RTCEEx Firmware driver defines

37.3.1 RTCEEx

Add 1 Second Parameter Definitions

RTC_SHIFTADD1S_RESET

RTC_SHIFTADD1S_SET

IS_RTC_SHIFT_ADD1S

Backup Registers Definitions

RTC_BKP_DR0

RTC_BKP_DR1

RTC_BKP_DR2

RTC_BKP_DR3

RTC_BKP_DR4

RTC_BKP_DR5
RTC_BKP_DR6
RTC_BKP_DR7
RTC_BKP_DR8
RTC_BKP_DR9
RTC_BKP_DR10
RTC_BKP_DR11
RTC_BKP_DR12
RTC_BKP_DR13
RTC_BKP_DR14
RTC_BKP_DR15
RTC_BKP_DR16
RTC_BKP_DR17
RTC_BKP_DR18
RTC_BKP_DR19
RTC_BKP_DR20
RTC_BKP_DR21
RTC_BKP_DR22
RTC_BKP_DR23
RTC_BKP_DR24
RTC_BKP_DR25
RTC_BKP_DR26
RTC_BKP_DR27
RTC_BKP_DR28
RTC_BKP_DR29
RTC_BKP_DR30
RTC_BKP_DR31
IS_RTC_BKP

Calib Output Selection Definitions

RTC_CALIBOUTPUT_512HZ
RTC_CALIBOUTPUT_1HZ
IS_RTC_CALIB_OUTPUT

Digital Calibration Definitions

RTC_CALIBSIGN_POSITIVE
RTC_CALIBSIGN_NEGATIVE
IS_RTC_CALIB_SIGN

IS_RTC_CALIB_VALUE**RTCEx Exported Macros****_HAL_RTC_WAKEUPTIMER_ENABLE****Description:**

- Enable the RTC WakeUp Timer peripheral.

Parameters:

- `_HANDLE_`: specifies the RTC handle.

Return value:

- None

Description:

- Enable the RTC TimeStamp peripheral.

Parameters:

- `_HANDLE_`: specifies the RTC handle.

Return value:

- None

Description:

- Disable the RTC WakeUp Timer peripheral.

Parameters:

- `_HANDLE_`: specifies the RTC handle.

Return value:

- None

Description:

- Disable the RTC TimeStamp peripheral.

Parameters:

- `_HANDLE_`: specifies the RTC handle.

Return value:

- None

Description:

- Enable the Coarse calibration process.

Parameters:

- `_HANDLE_`: specifies the

RTC handle.

Return value:

- None

`__HAL_RTC_COARSE_CALIB_DISABLE`

Description:

- Disable the Coarse calibration process.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`__HAL_RTC_CALIBRATION_OUTPUT_ENABLE`

Description:

- Enable the RTC calibration output.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`__HAL_RTC_CALIBRATION_OUTPUT_DISABLE`

Description:

- Disable the calibration output.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`__HAL_RTC_CLOCKREF_DETECTION_ENABLE`

Description:

- Enable the clock reference detection.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

`__HAL_RTC_CLOCKREF_DETECTION_DISABLE`

Description:

- Disable the clock reference detection.

Parameters:

- `__HANDLE__`: specifies the

RTC handle.

Return value:

- None

`_HAL_RTC_TIMESTAMP_ENABLE_IT`

Description:

- Enable the RTC TimeStamp interrupt.

Parameters:

- `_HANDLE_`: specifies the RTC handle.
- `_INTERRUPT_`: specifies the RTC TimeStamp interrupt sources to be enabled or disabled. This parameter can be:
 - `RTC_IT_TS`: TimeStamp interrupt

Return value:

- None

`_HAL_RTC_WAKEUPTIMER_ENABLE_IT`

Description:

- Enable the RTC WakeUpTimer interrupt.

Parameters:

- `_HANDLE_`: specifies the RTC handle.
- `_INTERRUPT_`: specifies the RTC WakeUpTimer interrupt sources to be enabled or disabled. This parameter can be:
 - `RTC_IT_WUT`: WakeUpTimer A interrupt

Return value:

- None

`_HAL_RTC_TIMESTAMP_DISABLE_IT`

Description:

- Disable the RTC TimeStamp interrupt.

Parameters:

- `_HANDLE_`: specifies the RTC handle.
- `_INTERRUPT_`: specifies the RTC TimeStamp interrupt sources to be enabled or disabled. This parameter can be:

- RTC_IT_TS: TimeStamp interrupt

Return value:

- None

`_HAL_RTC_WAKEUPTIMER_DISABLE_IT`

- Description:**
- Disable the RTC WakeUpTimer interrupt.

Parameters:

- `_HANDLE_`: specifies the RTC handle.
- `_INTERRUPT_`: specifies the RTC WakeUpTimer interrupt sources to be enabled or disabled. This parameter can be:
 - RTC_IT_WUT: WakeUpTimer A interrupt

Return value:

- None

`_HAL_RTC_TAMPER1_ENABLE`

- Description:**
- Enable the RTC Tamper1 input detection.

Parameters:

- `_HANDLE_`: specifies the RTC handle.

Return value:

- None

`_HAL_RTC_TAMPER1_DISABLE`

- Description:**
- Disable the RTC Tamper1 input detection.

Parameters:

- `_HANDLE_`: specifies the RTC handle.

Return value:

- None

`_HAL_RTC_TAMPER2_ENABLE`

- Description:**
- Enable the RTC Tamper2 input detection.

Parameters:

- `_HANDLE_`: specifies the RTC handle.

Return value:

- None

Description:

- Disable the RTC Tamper2 input detection.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

Description:

- Enable the RTC Tamper3 input detection.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

Description:

- Disable the RTC Tamper3 input detection.

Parameters:

- `__HANDLE__`: specifies the RTC handle.

Return value:

- None

Description:

- Check whether the specified RTC Tamper interrupt has occurred or not.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__INTERRUPT__`: specifies the RTC Tamper interrupt to check. This parameter can be:
 - `RTC_IT_TAMP1`: Tamper1 interrupt
 - `RTC_IT_TAMP2`: Tamper2 interrupt
 - `RTC_IT_TAMP3`:

Tamper3 interrupt

Return value:

- None

`__HAL_RTC_TAMPER_ENABLE_IT`

Description:

- Enable the RTC Tamper interrupt.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__INTERRUPT__`: specifies the RTC Tamper interrupt sources to be enabled. This parameter can be any combination of the following values:
 - `RTC_IT_TAMP1`: Tamper1 interrupt
 - `RTC_IT_TAMP2`: Tamper2 interrupt (*)
 - `RTC_IT_TAMP3`: Tamper3 interrupt (*)

Return value:

- None

Notes:

- (*) Available only on devices STM32L100xBA, STM32L151xBA, STM32L152xBA, STM32L100xC, STM32L151xC, STM32L152xC, STM32L162xC, STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD, STM32L151xE, STM32L152xE, STM32L162xE STM32L151xDX, STM32L152xDX, STM32L162xDX

`__HAL_RTC_TAMPER_DISABLE_IT`

Description:

- Disable the RTC Tamper interrupt.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__INTERRUPT__`: specifies the RTC Tamper interrupt sources to be disabled. This parameter can be any combination of the following values:
 - `RTC_IT_TAMP1`: Tamper1 interrupt
 - `RTC_IT_TAMP2`: Tamper2 interrupt (*)
 - `RTC_IT_TAMP3`: Tamper3 interrupt (*)

Return value:

- None

Notes:

- (*) Available only on devices STM32L100xBA, STM32L151xBA, STM32L152xBA, STM32L100xC, STM32L151xC, STM32L152xC, STM32L162xC, STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD, STM32L151xE, STM32L152xE, STM32L162xE, STM32L151xDX, STM32L152xDX, STM32L162xDX

`__HAL_RTC_TAMPER_GET_IT_SOURCE`**Description:**

- Check whether the specified RTC Tamper interrupt has been enabled or not.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__INTERRUPT__`: specifies the RTC Tamper interrupt source to check. This parameter can be:
 - `RTC_IT_TAMP1`: Tamper1 interrupt
 - `RTC_IT_TAMP2`:

- Tamper2 interrupt (*)
- RTC_IT_TAMP3:
Tamper3 interrupt (*)

Return value:

- None

Notes:

- (*) Available only on devices
STM32L100xBA,
STM32L151xBA,
STM32L152xBA,
STM32L100xC,
STM32L151xC,
STM32L152xC,
STM32L162xC,
STM32L151xCA,
STM32L151xD,
STM32L152xCA,
STM32L162xD,
STM32L151xE,
STM32L152xE,
STM32L162xE
STM32L151xDX,
STM32L152xDX,
STM32L162xDX

_HAL_RTC_WAKEUPTIMER_GET_IT**Description:**

- Check whether the specified RTC WakeUpTimer interrupt has occurred or not.

Parameters:

- _HANDLE_: specifies the RTC handle.
- _FLAG_: specifies the RTC WakeUpTimer interrupt sources to be enabled or disabled. This parameter can be:
 - RTC_IT_WUT:
WakeUpTimer A interrupt

Return value:

- None

_HAL_RTC_WAKEUPTIMER_GET_IT_SOURCE**Description:**

- Check whether the specified RTC Wake Up timer interrupt has been enabled or not.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__INTERRUPT__`: specifies the RTC Wake Up timer interrupt sources to check. This parameter can be:
 - `RTC_IT_WUT`: WakeUpTimer interrupt

Return value:

- None

Description:

- Check whether the specified RTC TimeStamp interrupt has occurred or not.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__INTERRUPT__`: specifies the RTC TimeStamp interrupt sources to be enabled or disabled. This parameter can be:
 - `RTC_IT_TS`: TimeStamp interrupt

Return value:

- None

Description:

- Check whether the specified RTC Time Stamp interrupt has been enabled or not.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__INTERRUPT__`: specifies the RTC Time Stamp interrupt source to check. This parameter can be:
 - `RTC_IT_TS`: TimeStamp interrupt

Return value:

- None

Description:

- Get the selected RTC TimeStamp's flag status.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__FLAG__`: specifies the RTC TimeStamp Flag sources to be enabled or disabled. This parameter can be:
 - `RTC_FLAG_TSF`
 - `RTC_FLAG_TSOVF`

Return value:

- None

`__HAL_RTC_WAKEUPTIMER_GET_FLAG`**Description:**

- Get the selected RTC WakeUpTimer's flag status.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__FLAG__`: specifies the RTC WakeUpTimer Flag sources to be enabled or disabled. This parameter can be:
 - `RTC_FLAG_WUTF`
 - `RTC_FLAG_WUTWF`

Return value:

- None

`__HAL_RTC_TAMPER_GET_FLAG`**Description:**

- Get the selected RTC Tamper's flag status.

Parameters:

- `__HANDLE__`: specifies the RTC handle.
- `__FLAG__`: specifies the RTC Tamper Flag sources to be enabled or disabled. This parameter can be:
 - `RTC_FLAG_TAMP1F`
 - `RTC_FLAG_TAMP2F (*)`
 - `RTC_FLAG_TAMP3F (*)`

Return value:

- None

Notes:

- (*) Available only on devices STM32L100xBA, STM32L151xBA, STM32L152xBA, STM32L100xC, STM32L151xC,

STM32L152xC,
STM32L162xC,
STM32L151xCA,
STM32L151xD,
STM32L152xCA,
STM32L152xD,
STM32L162xCA,
STM32L162xD,
STM32L151xE,
STM32L152xE,
STM32L162xE
STM32L151xDX,
STM32L152xDX,
STM32L162xDX

_HAL_RTC_SHIFT_GET_FLAG

Description:

- Get the selected RTC shift operation's flag status.

Parameters:

- _HANDLE_: specifies the RTC handle.
- _FLAG_: specifies the RTC shift operation Flag is pending or not. This parameter can be:
 - RTC_FLAG_SHPF

Return value:

- None

_HAL_RTC_TIMESTAMP_CLEAR_FLAG

Description:

- Clear the RTC Time Stamp's pending flags.

Parameters:

- _HANDLE_: specifies the RTC handle.
- _FLAG_: specifies the RTC Alarm Flag sources to be enabled or disabled. This parameter can be:
 - RTC_FLAG_TSF

Return value:

- None

_HAL_RTC_TAMPER_CLEAR_FLAG

Description:

- Clear the RTC Tamper's pending flags.

Parameters:

- _HANDLE_: specifies the RTC handle.
- _FLAG_: specifies the RTC

Tamper Flag sources to be enabled or disabled. This parameter can be:

- RTC_FLAG_TAMP1F
- RTC_FLAG_TAMP2F (*)
- RTC_FLAG_TAMP3F (*)

Return value:

- None

Notes:

- (*) Available only on devices STM32L100xBA, STM32L151xBA, STM32L152xBA, STM32L100xC, STM32L151xC, STM32L152xC, STM32L162xC, STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD, STM32L151xE, STM32L152xE, STM32L162xE, STM32L151xDX, STM32L152xDX, STM32L162xDX

[__HAL_RTC_WAKEUPTIMER_CLEAR_FLAG](#)

Description:

- Clear the RTC Wake Up timer's pending flags.

Parameters:

- [__HANDLE__](#): specifies the RTC handle.
- [__FLAG__](#): specifies the RTC WakeUpTimer Flag to clear. This parameter can be:
 - RTC_FLAG_WUTF

Return value:

- None

[__HAL_RTC_WAKEUPTIMER_EXTI_ENABLE_IT](#)

Description:

- Enable interrupt on the RTC WakeUp Timer associated Exti line.

Return value:

- None

`__HAL_RTC_WAKEUPTIMER_EXTI_DISABLE_IT`

Description:

- Disable interrupt on the RTC WakeUp Timer associated Exti line.

Return value:

- None

`__HAL_RTC_WAKEUPTIMER_EXTI_ENABLE_EVENT`

Description:

- Enable event on the RTC WakeUp Timer associated Exti line.

Return value:

- None.

`__HAL_RTC_WAKEUPTIMER_EXTI_DISABLE_EVENT`

Description:

- Disable event on the RTC WakeUp Timer associated Exti line.

Return value:

- None.

`__HAL_RTC_WAKEUPTIMER_EXTI_ENABLE_FALLING_EDGE`

Description:

- Enable falling edge trigger on the RTC WakeUp Timer associated Exti line.

Return value:

- None.

`__HAL_RTC_WAKEUPTIMER_EXTI_DISABLE_FALLING_EDGE`

Description:

- Disable falling edge trigger on the RTC WakeUp Timer associated Exti line.

Return value:

- None.

`__HAL_RTC_WAKEUPTIMER_EXTI_ENABLE_RISING_EDGE`

Description:

- Enable rising edge trigger on the RTC WakeUp Timer associated Exti line.

Return value:

- None.

`__HAL_RTC_WAKEUPTIMER_EXTI_DISABLE_RISING_EDGE`

Description:

- Disable rising edge trigger on the RTC WakeUp Timer associated Exti line.

Return value:

`__HAL_RTC_WAKEUPTIMER_EXTI_ENABLE_RISING_FALLING_EDGE`

- None.

Description:

- Enable rising & falling edge trigger on the RTC WakeUp Timer associated Exti line.

Return value:

- None.

`__HAL_RTC_WAKEUPTIMER_EXTI_DISABLE_RISING_FALLING_EDGE`

Description:

- Disable rising & falling edge trigger on the RTC WakeUp Timer associated Exti line.

Return value:

- None.

`__HAL_RTC_WAKEUPTIMER_EXTI_GET_FLAG`

- Check whether the RTC WakeUp Timer associated Exti line interrupt flag is set or not.

Return value:

- Line: Status.

`__HAL_RTC_WAKEUPTIMER_EXTI_CLEAR_FLAG`

- Clear the RTC WakeUp Timer associated Exti line flag.

Return value:

- None.

`__HAL_RTC_WAKEUPTIMER_EXTI_GENERATE_SWIT`

Description:

- Generate a Software interrupt on the RTC WakeUp Timer associated Exti line.

Return value:

- None.

`__HAL_RTC_TAMPER_TIMESTAMP_EXTI_ENABLE_IT`

Description:

- Enable interrupt on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None

`__HAL_RTC_TAMPER_TIMESTAMP_EXTI_DISABLE_IT`

Description:

- Disable interrupt on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None

Description:

- Enable event on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None.

Description:

- Disable event on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None.

Description:

- Enable falling edge trigger on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None.

Description:

- Disable falling edge trigger on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None.

Description:

- Enable rising edge trigger on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None.

Description:

- Disable rising edge trigger on the RTC Tamper and Timestamp associated Exti line.

Return value:

`__HAL_RTC_TAMPER_TIMESTAMP_EXTI_ENABLE_
RISING_FALLING_EDGE`

- None.

Description:

- Enable rising & falling edge trigger on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None.

`__HAL_RTC_TAMPER_TIMESTAMP_EXTI_DISABLE_
RISING_FALLING_EDGE`

Description:

- Disable rising & falling edge trigger on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None.

`__HAL_RTC_TAMPER_TIMESTAMP_EXTI_GET_FLAG`

Description:

- Check whether the RTC Tamper and Timestamp associated Exti line interrupt flag is set or not.

Return value:

- Line: Status.

`__HAL_RTC_TAMPER_TIMESTAMP_EXTI_CLEAR_FLAG`

Description:

- Clear the RTC Tamper and Timestamp associated Exti line flag.

Return value:

- None.

`__HAL_RTC_TAMPER_TIMESTAMP_EXTI_GENERATE_
SWIT`

Description:

- Generate a Software interrupt on the RTC Tamper and Timestamp associated Exti line.

Return value:

- None.

Output selection Definitions

RTC_OUTPUT_DISABLE
 RTC_OUTPUT_ALARMA
 RTC_OUTPUT_ALARMB
 RTC_OUTPUT_WAKEUP
 IS_RTC_OUTPUT

Smooth Calib Minus Pulses Definitions

IS_RTC_SMOOTH_CALIB_MINUS

Smooth Calib Period Definitions

RTC_SMOOTHCALIB_PERIOD_32SEC	If RTCCLK = 32768 Hz, Smooth calibration period is 32s, else $2^{\text{exp}20}$ RTCCLK seconds
RTC_SMOOTHCALIB_PERIOD_16SEC	If RTCCLK = 32768 Hz, Smooth calibration period is 16s, else $2^{\text{exp}19}$ RTCCLK seconds
RTC_SMOOTHCALIB_PERIOD_8SEC	If RTCCLK = 32768 Hz, Smooth calibration period is 8s, else $2^{\text{exp}18}$ RTCCLK seconds

IS_RTC_SMOOTH_CALIB_PERIOD

Smooth Calib Plus Pulses Definitions

RTC_SMOOTHCALIB_PLUSPULSES_SET	The number of RTCCLK pulses added during a X -second window = Y - CALM[8:0] with Y = 512, 256, 128 when X = 32, 16, 8
RTC_SMOOTHCALIB_PLUSPULSES_RESET	The number of RTCCLK pulses substituted during a 32-second window = CALM[8:0]

IS_RTC_SMOOTH_CALIB_PLUS

Subtract Fraction Of Second Value

IS_RTC_SHIFT_SUBFS

Tamper Filter Definitions

RTC_TAMPERFILTER_DISABLE	Tamper filter is disabled
RTC_TAMPERFILTER_2SAMPLE	Tamper is activated after 2 consecutive samples at the active level
RTC_TAMPERFILTER_4SAMPLE	Tamper is activated after 4 consecutive samples at the active level
RTC_TAMPERFILTER_8SAMPLE	Tamper is activated after 8 consecutive samples at the active level.

IS_RTC_TAMPER_FILTER

Tamper Pins Definitions

RTC_TAMPER_1
 RTC_TAMPER_2
 RTC_TAMPER_3
 IS_RTC_TAMPER

Tamper Pin Precharge Duration

RTC_TAMPERPRECHARGEDURATION_1RTCCLK	Tamper pins are pre-charged before sampling during 1 RTCCLK cycle
RTC_TAMPERPRECHARGEDURATION_2RTCCLK	Tamper pins are pre-charged before sampling during 2 RTCCLK cycles
RTC_TAMPERPRECHARGEDURATION_4RTCCLK	Tamper pins are pre-charged before sampling during 4 RTCCLK cycles
RTC_TAMPERPRECHARGEDURATION_8RTCCLK	Tamper pins are pre-charged before sampling during 8 RTCCLK cycles

IS_RTC_TAMPER_PRECHARGE_DURATION

Tamper Pull-Up Definitions

RTC_TAMPER_PULLUP_ENABLE	TimeStamp on Tamper Detection event saved
RTC_TAMPER_PULLUP_DISABLE	TimeStamp on Tamper Detection event is not saved

IS_RTC_TAMPER_PULLUP_STATE

Tamper Sampling Frequencies

RTC_TAMPERSAMPLINGFREQ_RTCCLK_DIV32768	Each of the tamper inputs are sampled with a frequency = RTCCLK / 32768
RTC_TAMPERSAMPLINGFREQ_RTCCLK_DIV16384	Each of the tamper inputs are sampled with a frequency = RTCCLK / 16384
RTC_TAMPERSAMPLINGFREQ_RTCCLK_DIV8192	Each of the tamper inputs are sampled with a frequency = RTCCLK / 8192
RTC_TAMPERSAMPLINGFREQ_RTCCLK_DIV4096	Each of the tamper inputs are sampled with a frequency = RTCCLK / 4096
RTC_TAMPERSAMPLINGFREQ_RTCCLK_DIV2048	Each of the tamper inputs are sampled with a frequency = RTCCLK / 2048
RTC_TAMPERSAMPLINGFREQ_RTCCLK_DIV1024	Each of the tamper inputs are sampled with a frequency = RTCCLK / 1024
RTC_TAMPERSAMPLINGFREQ_RTCCLK_DIV512	Each of the tamper inputs are sampled with a frequency = RTCCLK / 512
RTC_TAMPERSAMPLINGFREQ_RTCCLK_DIV256	Each of the tamper inputs are sampled with a frequency = RTCCLK / 256

IS_RTC_TAMPER_SAMPLING_FREQ

TimeStampOnTamperDetection Definitions

RTC_TIMESTAMPONTAMPERDETECTION_ENABLE

TimeStamp on Tamper
Detection event saved

RTC_TIMESTAMPONTAMPERDETECTION_DISABLE

TimeStamp on Tamper
Detection event is not saved

IS_RTC_TAMPER_TIMESTAMPONTAMPER_DETECTION

Tamper Trigger Definitions

RTC_TAMPERTRIGGER_RISINGEDGE

RTC_TAMPERTRIGGER_FALLINGEDGE

RTC_TAMPERTRIGGER_LOWLEVEL

RTC_TAMPERTRIGGER_HIGHLEVEL

IS_RTC_TAMPER_TRIGGER

Time Stamp Edges Definitions

RTC_TIMESTAMPEDGE_RISING

RTC_TIMESTAMPEDGE_FALLING

IS_TIMESTAMP_EDGE

Wakeup Timer Definitions

RTC_WAKEUPCLOCK_RTCCLK_DIV16

RTC_WAKEUPCLOCK_RTCCLK_DIV8

RTC_WAKEUPCLOCK_RTCCLK_DIV4

RTC_WAKEUPCLOCK_RTCCLK_DIV2

RTC_WAKEUPCLOCK_CK_SPRE_16BITS

RTC_WAKEUPCLOCK_CK_SPRE_17BITS

IS_RTC_WAKEUP_CLOCK

IS_RTC_WAKEUP_COUNTER

38 HAL SD Generic Driver

38.1 SD Firmware driver registers structures

38.1.1 SD_HandleTypeDef

Data Fields

- *SD_TypeDef * Instance*
- *SD_InitTypeDef Init*
- *HAL_LockTypeDef Lock*
- *uint32_t CardType*
- *uint32_t RCA*
- *uint32_t CSD*
- *uint32_t CID*
- *_IO uint32_t SdTransferCplt*
- *_IO uint32_t SdTransferErr*
- *_IO uint32_t DmaTransferCplt*
- *_IO uint32_t SdOperation*
- *DMA_HandleTypeDef * hdmarx*
- *DMA_HandleTypeDef * hdmatx*

Field Documentation

- ***SD_TypeDef* SD_HandleTypeDef::Instance***
SDIO register base address
- ***SD_InitTypeDef SD_HandleTypeDef::Init***
SD required parameters
- ***HAL_LockTypeDef SD_HandleTypeDef::Lock***
SD locking object
- ***uint32_t SD_HandleTypeDef::CardType***
SD card type
- ***uint32_t SD_HandleTypeDef::RCA***
SD relative card address
- ***uint32_t SD_HandleTypeDef::CSD[4]***
SD card specific data table
- ***uint32_t SD_HandleTypeDef::CID[4]***
SD card identification number table
- ***_IO uint32_t SD_HandleTypeDef::SdTransferCplt***
SD transfer complete flag in non blocking mode
- ***_IO uint32_t SD_HandleTypeDef::SdTransferErr***
SD transfer error flag in non blocking mode
- ***_IO uint32_t SD_HandleTypeDef::DmaTransferCplt***
SD DMA transfer complete flag
- ***_IO uint32_t SD_HandleTypeDef::SdOperation***
SD transfer operation (read/write)
- ***DMA_HandleTypeDef* SD_HandleTypeDef::hdmarx***
SD Rx DMA handle parameters
- ***DMA_HandleTypeDef* SD_HandleTypeDef::hdmatx***
SD Tx DMA handle parameters

38.1.2 HAL_SD_CSDTypedef

Data Fields

- `__IO uint8_t CSDStruct`
- `__IO uint8_t SysSpecVersion`
- `__IO uint8_t Reserved1`
- `__IO uint8_t TAAC`
- `__IO uint8_t NSAC`
- `__IO uint8_t MaxBusClkFrec`
- `__IO uint16_t CardComdClasses`
- `__IO uint8_t RdBlockLen`
- `__IO uint8_t PartBlockRead`
- `__IO uint8_t WrBlockMisalign`
- `__IO uint8_t RdBlockMisalign`
- `__IO uint8_t DSRImpl`
- `__IO uint8_t Reserved2`
- `__IO uint32_t DeviceSize`
- `__IO uint8_t MaxRdCurrentVDDMin`
- `__IO uint8_t MaxRdCurrentVDDMax`
- `__IO uint8_t MaxWrCurrentVDDMin`
- `__IO uint8_t MaxWrCurrentVDDMax`
- `__IO uint8_t DeviceSizeMul`
- `__IO uint8_t EraseGrSize`
- `__IO uint8_t EraseGrMul`
- `__IO uint8_t WrProtectGrSize`
- `__IO uint8_t WrProtectGrEnable`
- `__IO uint8_t ManDeflECC`
- `__IO uint8_t WrSpeedFact`
- `__IO uint8_t MaxWrBlockLen`
- `__IO uint8_t WriteBlockPaPartial`
- `__IO uint8_t Reserved3`
- `__IO uint8_t ContentProtectAppli`
- `__IO uint8_t FileFormatGrouop`
- `__IO uint8_t CopyFlag`
- `__IO uint8_t PermWrProtect`
- `__IO uint8_t TempWrProtect`
- `__IO uint8_t FileFormat`
- `__IO uint8_t ECC`
- `__IO uint8_t CSD_CRC`
- `__IO uint8_t Reserved4`

Field Documentation

- `__IO uint8_t HAL_SD_CSDTypedef::CSDStruct`
CSD structure
- `__IO uint8_t HAL_SD_CSDTypedef::SysSpecVersion`
System specification version
- `__IO uint8_t HAL_SD_CSDTypedef::Reserved1`
Reserved
- `__IO uint8_t HAL_SD_CSDTypedef::TAAC`
Data read access time 1
- `__IO uint8_t HAL_SD_CSDTypedef::NSAC`
Data read access time 2 in CLK cycles

- `__IO uint8_t HAL_SD_CSDTypeDef::MaxBusClkFrec`
Max. bus clock frequency
- `__IO uint16_t HAL_SD_CSDTypeDef::CardComdClasses`
Card command classes
- `__IO uint8_t HAL_SD_CSDTypeDef::RdBlockLen`
Max. read data block length
- `__IO uint8_t HAL_SD_CSDTypeDef::PartBlockRead`
Partial blocks for read allowed
- `__IO uint8_t HAL_SD_CSDTypeDef::WrBlockMisalign`
Write block misalignment
- `__IO uint8_t HAL_SD_CSDTypeDef::RdBlockMisalign`
Read block misalignment
- `__IO uint8_t HAL_SD_CSDTypeDef::DSRImpl`
DSR implemented
- `__IO uint8_t HAL_SD_CSDTypeDef::Reserved2`
Reserved
- `__IO uint32_t HAL_SD_CSDTypeDef::DeviceSize`
Device Size
- `__IO uint8_t HAL_SD_CSDTypeDef::MaxRdCurrentVDDMin`
Max. read current @ VDD min
- `__IO uint8_t HAL_SD_CSDTypeDef::MaxRdCurrentVDDMax`
Max. read current @ VDD max
- `__IO uint8_t HAL_SD_CSDTypeDef::MaxWrCurrentVDDMin`
Max. write current @ VDD min
- `__IO uint8_t HAL_SD_CSDTypeDef::MaxWrCurrentVDDMax`
Max. write current @ VDD max
- `__IO uint8_t HAL_SD_CSDTypeDef::DeviceSizeMul`
Device size multiplier
- `__IO uint8_t HAL_SD_CSDTypeDef::EraseGrSize`
Erase group size
- `__IO uint8_t HAL_SD_CSDTypeDef::EraseGrMul`
Erase group size multiplier
- `__IO uint8_t HAL_SD_CSDTypeDef::WrProtectGrSize`
Write protect group size
- `__IO uint8_t HAL_SD_CSDTypeDef::WrProtectGrEnable`
Write protect group enable
- `__IO uint8_t HAL_SD_CSDTypeDef::ManDefIECC`
Manufacturer default ECC
- `__IO uint8_t HAL_SD_CSDTypeDef::WrSpeedFact`
Write speed factor
- `__IO uint8_t HAL_SD_CSDTypeDef::MaxWrBlockLen`
Max. write data block length
- `__IO uint8_t HAL_SD_CSDTypeDef::WriteBlockPaPartial`
Partial blocks for write allowed
- `__IO uint8_t HAL_SD_CSDTypeDef::Reserved3`
Reserved
- `__IO uint8_t HAL_SD_CSDTypeDef::ContentProtectAppl`
Content protection application
- `__IO uint8_t HAL_SD_CSDTypeDef::FileFormatGrouop`
File format group
- `__IO uint8_t HAL_SD_CSDTypeDef::CopyFlag`
Copy flag (OTP)
- `__IO uint8_t HAL_SD_CSDTypeDef::PermWrProtect`
Permanent write protection

- `__IO uint8_t HAL_SD_CSDTypedef::TempWrProtect`
Temporary write protection
- `__IO uint8_t HAL_SD_CSDTypedef::FileFormat`
File format
- `__IO uint8_t HAL_SD_CSDTypedef::ECC`
ECC code
- `__IO uint8_t HAL_SD_CSDTypedef::CSD_CRC`
CSD CRC
- `__IO uint8_t HAL_SD_CSDTypedef::Reserved4`
Always 1

38.1.3 HAL_SD_CIDTypedef

Data Fields

- `__IO uint8_t ManufacturerID`
- `__IO uint16_t OEM_AppId`
- `__IO uint32_t ProdName1`
- `__IO uint8_t ProdName2`
- `__IO uint8_t ProdRev`
- `__IO uint32_t ProdSN`
- `__IO uint8_t Reserved1`
- `__IO uint16_t ManufactDate`
- `__IO uint8_t CID_CRC`
- `__IO uint8_t Reserved2`

Field Documentation

- `__IO uint8_t HAL_SD_CIDTypedef::ManufacturerID`
Manufacturer ID
- `__IO uint16_t HAL_SD_CIDTypedef::OEM_AppId`
OEM/Application ID
- `__IO uint32_t HAL_SD_CIDTypedef::ProdName1`
Product Name part1
- `__IO uint8_t HAL_SD_CIDTypedef::ProdName2`
Product Name part2
- `__IO uint8_t HAL_SD_CIDTypedef::ProdRev`
Product Revision
- `__IO uint32_t HAL_SD_CIDTypedef::ProdSN`
Product Serial Number
- `__IO uint8_t HAL_SD_CIDTypedef::Reserved1`
Reserved1
- `__IO uint16_t HAL_SD_CIDTypedef::ManufactDate`
Manufacturing Date
- `__IO uint8_t HAL_SD_CIDTypedef::CID_CRC`
CID CRC
- `__IO uint8_t HAL_SD_CIDTypedef::Reserved2`
Always 1

38.1.4 HAL_SD_CardStatusTypedef

Data Fields

- `__IO uint8_t DAT_BUS_WIDTH`
- `__IO uint8_t SECURED_MODE`
- `__IO uint16_t SD_CARD_TYPE`
- `__IO uint32_t SIZE_OF_PROTECTED_AREA`

- `__IO uint8_t SPEED_CLASS`
- `__IO uint8_t PERFORMANCE_MOVE`
- `__IO uint8_t AU_SIZE`
- `__IO uint16_t ERASE_SIZE`
- `__IO uint8_t ERASE_TIMEOUT`
- `__IO uint8_t ERASE_OFFSET`

Field Documentation

- `__IO uint8_t HAL_SD_CardStatusTypedef::DAT_BUS_WIDTH`
Shows the currently defined data bus width
- `__IO uint8_t HAL_SD_CardStatusTypedef::SECURED_MODE`
Card is in secured mode of operation
- `__IO uint16_t HAL_SD_CardStatusTypedef::SD_CARD_TYPE`
Carries information about card type
- `__IO uint32_t HAL_SD_CardStatusTypedef::SIZE_OF_PROTECTED_AREA`
Carries information about the capacity of protected area
- `__IO uint8_t HAL_SD_CardStatusTypedef::SPEED_CLASS`
Carries information about the speed class of the card
- `__IO uint8_t HAL_SD_CardStatusTypedef::PERFORMANCE_MOVE`
Carries information about the card's performance move
- `__IO uint8_t HAL_SD_CardStatusTypedef::AU_SIZE`
Carries information about the card's allocation unit size
- `__IO uint16_t HAL_SD_CardStatusTypedef::ERASE_SIZE`
Determines the number of AUs to be erased in one operation
- `__IO uint8_t HAL_SD_CardStatusTypedef::ERASE_TIMEOUT`
Determines the timeout for any number of AU erase
- `__IO uint8_t HAL_SD_CardStatusTypedef::ERASE_OFFSET`
Carries information about the erase offset

38.1.5 HAL_SD_CardInfoTypedef

Data Fields

- `HAL_SD_CSDTypedef SD_csd`
- `HAL_SD_CIDTypedef SD_cid`
- `uint64_t CardCapacity`
- `uint32_t CardBlockSize`
- `uint16_t RCA`
- `uint8_t CardType`

Field Documentation

- `HAL_SD_CSDTypedef HAL_SD_CardInfoTypedef::SD_csd`
SD card specific data register
- `HAL_SD_CIDTypedef HAL_SD_CardInfoTypedef::SD_cid`
SD card identification number register
- `uint64_t HAL_SD_CardInfoTypedef::CardCapacity`
Card capacity
- `uint32_t HAL_SD_CardInfoTypedef::CardBlockSize`
Card block size
- `uint16_t HAL_SD_CardInfoTypedef::RCA`
SD relative card address
- `uint8_t HAL_SD_CardInfoTypedef::CardType`
SD card type

38.2 SD Firmware driver API description

38.2.1 How to use this driver

This driver implements a high level communication layer for read and write from/to this memory. The needed STM32 hardware resources (SDIO and GPIO) are performed by the user in HAL_SD_MspInit() function (MSP layer). Basically, the MSP layer configuration should be the same as we provide in the examples. You can easily tailor this configuration according to hardware resources.

This driver is a generic layered driver for SDIO memories which uses the HAL SDIO driver functions to interface with SD and uSD cards devices. It is used as follows:

1. Initialize the SDIO low level resources by implement the HAL_SD_MspInit() API:
 - a. Enable the SDIO interface clock using __HAL_RCC_SDIO_CLK_ENABLE();
 - b. SDIO pins configuration for SD card
 - Enable the clock for the SDIO GPIOs using the functions
__HAL_RCC_GPIOx_CLK_ENABLE();
 - Configure these SDIO pins as alternate function pull-up using
HAL_GPIO_Init() and according to your pin assignment;
 - c. DMA Configuration if you need to use DMA process
(HAL_SD_ReadBlocks_DMA() and HAL_SD_WriteBlocks_DMA() APIs).
 - Enable the DMAx interface clock using
__HAL_RCC_DMAX_CLK_ENABLE();
 - Configure the DMA using the function HAL_DMA_Init() with predeclared and filled.
 - d. NVIC configuration if you need to use interrupt process when using DMA transfer.
 - Configure the SDIO and DMA interrupt priorities using functions
HAL_NVIC_SetPriority(); DMA priority is superior to SDIO's priority
 - Enable the NVIC DMA and SDIO IRQs using function
HAL_NVIC_EnableIRQ()
 - SDIO interrupts are managed using the macros
__HAL_SD_SDIO_ENABLE_IT() and __HAL_SD_SDIO_DISABLE_IT()
inside the communication process.
 - SDIO interrupts pending bits are managed using the macros
__HAL_SD_SDIO_GET_IT() and __HAL_SD_SDIO_CLEAR_IT()
2. At this stage, you can perform SD read/write/erase operations after SD card initialization

SD Card Initialization and configuration

To initialize the SD Card, use the HAL_SD_Init() function. It initializes the SD Card and put it into Standby State (Ready for data transfer). This function provide the following operations:

1. Apply the SD Card initialization process at 400KHz and check the SD Card type (Standard Capacity or High Capacity). You can change or adapt this frequency by adjusting the "ClockDiv" field. The SD Card frequency (SDIO_CK) is computed as follows: SDIO_CK = SDIOCLK / (ClockDiv + 2) In initialization mode and according to the SD Card standard, make sure that the SDIO_CK frequency doesn't exceed 400KHz.
2. Get the SD CID and CSD data. All these information are managed by the SDCardInfo structure. This structure provide also ready computed SD Card capacity and Block size. These information are stored in SD handle structure in case of future use.
3. Configure the SD Card Data transfer frequency. By Default, the card transfer frequency is set to 48MHz / (SDIO_TRANSFER_CLK_DIV + 2) = 8MHz. You can change or adapt this frequency by adjusting the "ClockDiv" field. The SD Card

frequency (SDIO_CK) is computed as follows: $SDIO_CK = SDIOCLK / (ClockDiv + 2)$. In transfer mode and according to the SD Card standard, make sure that the SDIO_CK frequency doesn't exceed 25MHz and 50MHz in High-speed mode switch. To be able to use a frequency higher than 24MHz, you should use the SDIO peripheral in bypass mode. Refer to the corresponding reference manual for more details.

4. Select the corresponding SD Card according to the address read with the step 2.
5. Configure the SD Card in wide bus mode: 4-bits data.

SD Card Read operation

- You can read from SD card in polling mode by using function `HAL_SD_ReadBlocks()`. This function support only 512-bytes block length (the block size should be chosen as 512 bytes). You can choose either one block read operation or multiple block read operation by adjusting the "NumberOfBlocks" parameter.
- You can read from SD card in DMA mode by using function `HAL_SD_ReadBlocks_DMA()`. This function support only 512-bytes block length (the block size should be chosen as 512 bytes). You can choose either one block read operation or multiple block read operation by adjusting the "NumberOfBlocks" parameter. After this, you have to call the function `HAL_SD_CheckReadOperation()`, to insure that the read transfer is done correctly in both DMA and SD sides.

SD Card Write operation

- You can write to SD card in polling mode by using function `HAL_SD_WriteBlocks()`. This function support only 512-bytes block length (the block size should be chosen as 512 bytes). You can choose either one block read operation or multiple block read operation by adjusting the "NumberOfBlocks" parameter.
- You can write to SD card in DMA mode by using function `HAL_SD_WriteBlocks_DMA()`. This function support only 512-bytes block length (the block size should be chosen as 512 byte). You can choose either one block read operation or multiple block read operation by adjusting the "NumberOfBlocks" parameter. After this, you have to call the function `HAL_SD_CheckWriteOperation()`, to insure that the write transfer is done correctly in both DMA and SD sides.

SD card status

- At any time, you can check the SD Card status and get the SD card state by using the `HAL_SD_GetStatus()` function. This function checks first if the SD card is still connected and then get the internal SD Card transfer state.
- You can also get the SD card SD Status register by using the `HAL_SD_SendSDStatus()` function.

SD HAL driver macros list



You can refer to the SD HAL driver header file for more useful macros

38.2.2 Initialization and de-initialization functions

This section provides functions allowing to initialize/de-initialize the SD card device to be ready for use.

This section contains the following APIs:

- [*HAL_SD_Init\(\)*](#)
- [*HAL_SD_DelInit\(\)*](#)
- [*HAL_SD_MspInit\(\)*](#)
- [*HAL_SD_MspDelInit\(\)*](#)

38.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the data transfer from/to SD card.

This section contains the following APIs:

- [*HAL_SD_ReadBlocks\(\)*](#)
- [*HAL_SD_WriteBlocks\(\)*](#)
- [*HAL_SD_ReadBlocks_DMA\(\)*](#)
- [*HAL_SD_WriteBlocks_DMA\(\)*](#)
- [*HAL_SD_CheckReadOperation\(\)*](#)
- [*HAL_SD_CheckWriteOperation\(\)*](#)
- [*HAL_SD_Erase\(\)*](#)
- [*HAL_SD_IRQHandler\(\)*](#)
- [*HAL_SD_XferCpltCallback\(\)*](#)
- [*HAL_SD_XferErrorCallback\(\)*](#)
- [*HAL_SD_DMA_RxCpltCallback\(\)*](#)
- [*HAL_SD_DMA_RxErrorCallback\(\)*](#)
- [*HAL_SD_DMA_TxCpltCallback\(\)*](#)
- [*HAL_SD_DMA_TxErrorCallback\(\)*](#)

38.2.4 Peripheral Control functions

This subsection provides a set of functions allowing to control the SD card operations.

This section contains the following APIs:

- [*HAL_SD_Get_CardInfo\(\)*](#)
- [*HAL_SD_WideBusOperation_Config\(\)*](#)
- [*HAL_SD_StopTransfer\(\)*](#)
- [*HAL_SD_HighSpeed\(\)*](#)

38.2.5 Peripheral State functions

This subsection permits to get in runtime the status of the peripheral and the data flow.

This section contains the following APIs:

- [*HAL_SD_SendSDStatus\(\)*](#)
- [*HAL_SD_GetStatus\(\)*](#)
- [*HAL_SD_GetCardStatus\(\)*](#)

38.2.6 Detailed description of functions

HAL_SD_Init

Function name **[*HAL_SD_ErrorTypeDef HAL_SD_Init \(SD_HandleTypeDef * hsd, HAL_SD_CardInfoTypeDef * SDCardInfo\)*](#)**

Function description Initializes the SD card according to the specified parameters in the **SD_HandleTypeDef** and create the associated handle.

Parameters

- **hsd:** SD handle

- **SDCardInfo:** HAL_SD_CardInfoTypedef structure for SD card information
 - **HAL:** SD error state
- Return values

HAL_SD_DelInit

Function name	HAL_StatusTypeDef HAL_SD_DelInit (SD_HandleTypeDef * hsd)
Function description	De-Initializes the SD card.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SD_MspInit

Function name	void HAL_SD_MspInit (SD_HandleTypeDef * hsd)
Function description	Initializes the SD MSP.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_SD_MspDelInit

Function name	void HAL_SD_MspDelInit (SD_HandleTypeDef * hsd)
Function description	De-Initialize SD MSP.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_SD_ReadBlocks

Function name	HAL_SD_ErrorTypeDef HAL_SD_ReadBlocks (SD_HandleTypeDef * hsd, uint32_t * pReadBuffer, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumberOfBlocks)
Function description	Reads block(s) from a specified address in a card.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle • pReadBuffer: pointer to the buffer that will contain the received data • ReadAddr: Address from where data is to be read • BlockSize: SD card Data block size • NumberOfBlocks: Number of SD blocks to read
Return values	<ul style="list-style-type: none"> • SD: Card error state
Notes	<ul style="list-style-type: none"> • BlockSize must be 512 bytes.

HAL_SD_WriteBlocks

Function name	HAL_SD_ErrorTypeDef HAL_SD_WriteBlocks (SD_HandleTypeDef * hsd, uint32_t * pWriteBuffer, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumberOfBlocks)
---------------	--

Function description	Allows to write block(s) to a specified address in a card.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle • pWriteBuffer: pointer to the buffer that will contain the data to transmit • WriteAddr: Address from where data is to be written • BlockSize: SD card Data block size • NumberOfBlocks: Number of SD blocks to write
Return values	<ul style="list-style-type: none"> • SD: Card error state
Notes	<ul style="list-style-type: none"> • BlockSize must be 512 bytes.

HAL_SD_Erase

Function name	HAL_SD_ErrorTypeDef HAL_SD_Erase (SD_HandleTypeDef * hsd, uint64_t startaddr, uint64_t endaddr)
Function description	Erases the specified memory area of the given SD card.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle • startaddr: Start byte address • endaddr: End byte address
Return values	<ul style="list-style-type: none"> • SD: Card error state

HAL_SD_IRQHandler

Function name	void HAL_SD_IRQHandler (SD_HandleTypeDef * hsd)
Function description	This function handles SD card interrupt request.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_SD_DMA_RxCpltCallback

Function name	void HAL_SD_DMA_RxCpltCallback (DMA_HandleTypeDef * hdma)
Function description	SD Transfer complete Rx callback in non blocking mode.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SD_DMA_RxErrorCallback

Function name	void HAL_SD_DMA_RxErrorCallback (DMA_HandleTypeDef * hdma)
Function description	SD DMA transfer complete Rx error callback.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module.

Return values	<ul style="list-style-type: none"> • None:
---------------	--

HAL_SD_DMA_TxCpltCallback

Function name	void HAL_SD_DMA_TxCpltCallback (DMA_HandleTypeDef * hdma)
Function description	SD Transfer complete Tx callback in non blocking mode.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SD_DMA_TxErrorCallback

Function name	void HAL_SD_DMA_TxErrorCallback (DMA_HandleTypeDef * hdma)
Function description	SD DMA transfer complete error Tx callback.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SD_XferCpltCallback

Function name	void HAL_SD_XferCpltCallback (SD_HandleTypeDef * hsd)
Function description	SD end of transfer callback.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_SD_XferErrorCallback

Function name	void HAL_SD_XferErrorCallback (SD_HandleTypeDef * hsd)
Function description	SD Transfer Error callback.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • None:

HAL_SD_ReadBlocks_DMA

Function name	HAL_SD_ErrorTypeDef HAL_SD_ReadBlocks_DMA (SD_HandleTypeDef * hsd, uint32_t * pReadBuffer, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumberOfBlocks)
Function description	Reads block(s) from a specified address in a card.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle • pReadBuffer: Pointer to the buffer that will contain the received data • ReadAddr: Address from where data is to be read • BlockSize: SD card Data block size

- **NumberOfBlocks:** Number of blocks to read.
- **SD:** Card error state
- This API should be followed by the function HAL_SD_CheckReadOperation() to check the completion of the read process
- BlockSize must be 512 bytes.

HAL_SD_WriteBlocks_DMA

- | | |
|----------------------|--|
| Function name | HAL_SD_ErrorTypeDef HAL_SD_WriteBlocks_DMA
(SD_HandleTypeDef * hsd, uint32_t * pWriteBuffer, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumberOfBlocks) |
| Function description | Writes block(s) to a specified address in a card. |
| Parameters | <ul style="list-style-type: none"> • hsd: SD handle • pWriteBuffer: pointer to the buffer that will contain the data to transmit • WriteAddr: Address from where data is to be read • BlockSize: the SD card Data block size • NumberOfBlocks: Number of blocks to write |
| Return values | • SD: Card error state |
| Notes | <ul style="list-style-type: none"> • This API should be followed by the function HAL_SD_CheckWriteOperation() to check the completion of the write process (by SD current status polling). • BlockSize must be 512 bytes. |

HAL_SD_CheckWriteOperation

- | | |
|----------------------|---|
| Function name | HAL_SD_ErrorTypeDef HAL_SD_CheckWriteOperation
(SD_HandleTypeDef * hsd, uint32_t Timeout) |
| Function description | This function waits until the SD DMA data write transfer is finished. |
| Parameters | <ul style="list-style-type: none"> • hsd: SD handle • Timeout: Timeout duration |
| Return values | • SD: Card error state |

HAL_SD_CheckReadOperation

- | | |
|----------------------|---|
| Function name | HAL_SD_ErrorTypeDef HAL_SD_CheckReadOperation
(SD_HandleTypeDef * hsd, uint32_t Timeout) |
| Function description | This function waits until the SD DMA data read transfer is finished. |
| Parameters | <ul style="list-style-type: none"> • hsd: SD handle • Timeout: Timeout duration |
| Return values | • SD: Card error state |

HAL_SD_Get_CardInfo

- | | |
|---------------|--|
| Function name | HAL_SD_ErrorTypeDef HAL_SD_Get_CardInfo
(SD_HandleTypeDef * hsd, HAL_SD_CardInfoTypeDef * |
|---------------|--|

pCardInfo)

Function description	Returns information about specific card.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle • pCardInfo: Pointer to a HAL_SD_CardInfoTypedef structure that contains all SD card information
Return values	<ul style="list-style-type: none"> • SD: Card error state

HAL_SD_WideBusOperation_Config

Function name	HAL_SD_ErrorTypedef HAL_SD_WideBusOperation_Config (SD_HandleTypeDef * hsd, uint32_t WideMode)
Function description	Enables wide bus operation for the requested card if supported by card.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle • WideMode: Specifies the SD card wide bus mode This parameter can be one of the following values: <ul style="list-style-type: none"> – SDIO_BUS_WIDE_8B: 8-bit data transfer (Only for MMC) – SDIO_BUS_WIDE_4B: 4-bit data transfer – SDIO_BUS_WIDE_1B: 1-bit data transfer
Return values	<ul style="list-style-type: none"> • SD: Card error state

HAL_SD_StopTransfer

Function name	HAL_SD_ErrorTypedef HAL_SD_StopTransfer (SD_HandleTypeDef * hsd)
Function description	Aborts an ongoing data transfer.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • SD: Card error state

HAL_SD_HighSpeed

Function name	HAL_SD_ErrorTypedef HAL_SD_HighSpeed (SD_HandleTypeDef * hsd)
Function description	Switches the SD card to High Speed mode.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • SD: Card error state

Notes

- This operation should be followed by the configuration of PLL to have SDIOCK clock between 67 and 75 MHz

HAL_SD_SendSDStatus

Function name	HAL_SD_ErrorTypedef HAL_SD_SendSDStatus (SD_HandleTypeDef * hsd, uint32_t * pSDstatus)
Function description	Returns the current SD card's status.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle

- | | |
|---------------|---|
| Return values | <ul style="list-style-type: none"> • pSDstatus: Pointer to the buffer that will contain the SD card status SD Status register) • SD: Card error state |
|---------------|---|

HAL_SD_GetCardStatus

Function name	HAL_SD_ErrorTypeDef HAL_SD_GetCardStatus (SD_HandleTypeDef * hsd, HAL_SD_CardStatusTypeDef * pCardStatus)
Function description	Gets the SD card status.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle • pCardStatus: Pointer to the HAL_SD_CardStatusTypeDef structure that will contain the SD card status information
Return values	<ul style="list-style-type: none"> • SD: Card error state

HAL_SD_GetStatus

Function name	HAL_SD_TransferStateTypeDef HAL_SD_GetStatus (SD_HandleTypeDef * hsd)
Function description	Gets the current sd card data status.
Parameters	<ul style="list-style-type: none"> • hsd: SD handle
Return values	<ul style="list-style-type: none"> • Data: Transfer state

38.3 SD Firmware driver defines

38.3.1 SD

SD Exported Constants

SD_CMD_GO_IDLE_STATE	Resets the SD memory card.
SD_CMD_SEND_OP_COND	Sends host capacity support information and activates the card's initialization process.
SD_CMD_ALL_SEND_CID	Asks any card connected to the host to send the CID numbers on the CMD line.
SD_CMD_SET_REL_ADDR	Asks the card to publish a new relative address (RCA).
SD_CMD_SET_DSR	Programs the DSR of all cards.
SD_CMD_SDIO_SEN_OP_COND	Sends host capacity support information (HCS) and asks the accessed card to send its operating condition register (OCR) content in the response on the CMD

SD_CMD_HS_SWITCH	Checks switchable function (mode 0) and switch card function (mode 1).
SD_CMD_SEL_DESEL_CARD	Selects the card by its own relative address and gets deselected by any other address
SD_CMD_HS_SEND_EXT_CSD	Sends SD Memory Card interface condition, which includes host supply voltage information and asks the card whether card supports voltage.
SD_CMD_SEND_CSD	Addressed card sends its card specific data (CSD) on the CMD line.
SD_CMD_SEND_CID	Addressed card sends its card identification (CID) on the CMD line.
SD_CMD_READ_DAT_UNTIL_STOP	SD card doesn't support it.
SD_CMD_STOP_TRANSMISSION	Forces the card to stop transmission.
SD_CMD_SEND_STATUS	Addressed card sends its status register.
SD_CMD_HS_BUSTEST_READ	Sends an addressed card into the inactive state.
SD_CMD_GO_INACTIVE_STATE	Sets the block length (in bytes for SDSC) for all following block commands (read, write, lock). Default block length is fixed to 512 Bytes. Not effective for SDHS and SDXC.
SD_CMD_SET_BLOCKLEN	Reads single block of size selected by SET_BLOCKLEN in case of SDSC, and a block of fixed 512 bytes in case of SDHC and SDXC.
SD_CMD_READ_SINGLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by STOP_TRANSMISSION command.
SD_CMD_READ_MULT_BLOCK	64 bytes tuning pattern is sent for SDR50 and
SD_CMD_HS_BUSTEST_WRITE	

SD_CMD_WRITE_DAT_UNTIL_STOP	SDR104. Speed class control command.
SD_CMD_SET_BLOCK_COUNT	Specify block count for CMD18 and CMD25.
SD_CMD_WRITE_SINGLE_BLOCK	Writes single block of size selected by SET_BLOCKLEN in case of SDSC, and a block of fixed 512 bytes in case of SDHC and SDXC.
SD_CMD_WRITE_MULT_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
SD_CMD_PROG_CID	Reserved for manufacturers.
SD_CMD_PROG_CSD	Programming of the programmable bits of the CSD.
SD_CMD_SET_WRITE_PROT	Sets the write protection bit of the addressed group.
SD_CMD_CLR_WRITE_PROT	Clears the write protection bit of the addressed group.
SD_CMD_SEND_WRITE_PROT	Asks the card to send the status of the write protection bits.
SD_CMD_SD_ERASE_GRP_START	Sets the address of the first write block to be erased. (For SD card only).
SD_CMD_SD_ERASE_GRP_END	Sets the address of the last write block of the continuous range to be erased.
SD_CMD_ERASE_GRP_START	Sets the address of the first write block to be erased. Reserved for each command system set by switch function command (CMD6).
SD_CMD_ERASE_GRP_END	Sets the address of the last write block of the continuous range to be erased. Reserved for each command system set by switch function command (CMD6).
SD_CMD_ERASE	Reserved for SD security applications.

SD_CMD_FAST_IO	SD card doesn't support it (Reserved).
SD_CMD_GO_IRQ_STATE	SD card doesn't support it (Reserved).
SD_CMD_LOCK_UNLOCK	Sets/resets the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
SD_CMD_APP_CMD	Indicates to the card that the next command is an application specific command rather than a standard command.
SD_CMD_GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose/application specific commands.
SD_CMD_NO_CMD	
SD_CMD_APP_SD_SET_BUSWIDTH	SDIO_APP_CMD should be sent before sending these commands. (ACMD6) Defines the data bus width to be used for data transfer. The allowed data bus widths are given in SCR register.
SD_CMD_SD_APP_STATUS	(ACMD13) Sends the SD status.
SD_CMD_SD_APP_SEND_NUM_WRITE_BLOCKS	(ACMD22) Sends the number of the written (without errors) write blocks. Responds with 32bit+CRC data block.
SD_CMD_SD_APP_OP_COND	(ACMD41) Sends host capacity support information (HCS) and asks the accessed card to send its operating condition register (OCR) content in the response on the CMD line.
SD_CMD_SD_APP_SET_CLR_CARD_DETECT	(ACMD42) Connects/Disconnects the 50 KOhm pull-up resistor on CD/DAT3 (pin 1) of the card.
SD_CMD_SD_APP_SEND_SCR	Reads the SD Configuration Register (SCR).
SD_CMD_SDIO_RW_DIRECT	For SD I/O card only,

	reserved for security specification.
SD_CMD_SDIO_RW_EXTENDED	For SD I/O card only, reserved for security specification.
SD_CMD_SD_APP_GET_MKB	SD_CMD_APP_CMD should be sent before sending these commands. For SD card only
SD_CMD_SD_APP_GET_MID	For SD card only
SD_CMD_SD_APP_SET_CER_RN1	For SD card only
SD_CMD_SD_APP_GET_CER_RN2	For SD card only
SD_CMD_SD_APP_SET_CER_RES2	For SD card only
SD_CMD_SD_APP_GET_CER_RES1	For SD card only
SD_CMD_SD_APP_SECURE_READ_MULTIPLE_BLOCK	For SD card only
SD_CMD_SD_APP_SECURE_WRITE_MULTIPLE_BLOCK	For SD card only
SD_CMD_SD_APP_SECURE_ERASE	For SD card only
SD_CMD_SD_APP_CHANGE_SECURE_AREA	For SD card only
SD_CMD_SD_APP_SECURE_WRITE_MKB	For SD card only
STD_CAPACITY_SD_CARD_V1_1	
STD_CAPACITY_SD_CARD_V2_0	
HIGH_CAPACITY_SD_CARD	
MULTIMEDIA_CARD	
SECURE_DIGITAL_IO_CARD	
HIGH_SPEED_MULTIMEDIA_CARD	
SECURE_DIGITAL_IO_COMBO_CARD	
HIGH_CAPACITY_MMC_CARD	

SD Exported Macros

`_HAL_SD_SDIO_ENABLE`

Description:

- Enable the SD device.

Return value:

- None

`_HAL_SD_SDIO_DISABLE`

Description:

- Disable the SD device.

Return value:

- None

`_HAL_SD_SDIO_DMA_ENABLE`

Description:

- Enable the SDIO DMA transfer.

Return value:

- None

[__HAL_SD_SDIO_DMA_DISABLE](#)

Description:

- Disable the SDIO DMA transfer.

Return value:

- None

[__HAL_SD_SDIO_ENABLE_IT](#)

Description:

- Enable the SD device interrupt.

Parameters:

- __HANDLE__: SD Handle
- __INTERRUPT__: specifies the SDIO interrupt sources to be enabled. This parameter can be one or a combination of the following values:
 - SDIO_IT_CCRCFAIL: Command response received (CRC check failed) interrupt
 - SDIO_IT_DCRCFAIL: Data block sent/received (CRC check failed) interrupt
 - SDIO_IT_CTIMEOUT: Command response timeout interrupt
 - SDIO_IT_DTIMEOUT: Data timeout interrupt
 - SDIO_IT_TXUNDERR: Transmit FIFO underrun error interrupt
 - SDIO_IT_RXOVERR: Received FIFO overrun error interrupt
 - SDIO_IT_CMDREND: Command response received (CRC check passed) interrupt
 - SDIO_IT_CMDSENT: Command sent (no response required) interrupt
 - SDIO_IT_DATAEND: Data end (data counter, SDIDCOUNT, is zero) interrupt
 - SDIO_IT_STBITERR: Start bit not detected on all data signals in wide bus mode interrupt
 - SDIO_IT_DBCKEND: Data block sent/received (CRC check passed) interrupt
 - SDIO_IT_CMDACT: Command transfer in progress interrupt
 - SDIO_IT_TXACT: Data transmit in progress interrupt
 - SDIO_IT_RXACT: Data receive in progress interrupt
 - SDIO_IT_TXFIFOHE: Transmit FIFO Half Empty interrupt
 - SDIO_IT_RXFIFOHF: Receive FIFO Half Full interrupt
 - SDIO_IT_TXFIFOF: Transmit FIFO full interrupt
 - SDIO_IT_RXFIFOF: Receive FIFO full interrupt

- interrupt
 - SDIO_IT_TXFIFOE: Transmit FIFO empty interrupt
 - SDIO_IT_RXFIFOE: Receive FIFO empty interrupt
 - SDIO_IT_TXDABL: Data available in transmit FIFO interrupt
 - SDIO_IT_RXDABL: Data available in receive FIFO interrupt
 - SDIO_IT_SDIOIT: SD I/O interrupt received interrupt
 - SDIO_IT_CEATAEND: CE-ATA command completion signal received for CMD61 interrupt

Return value:

- None

_HAL_SD_SDIO_DISABLE_IT

- Disable the SD device interrupt.

Parameters:

- _HANDLE_: SD Handle
- _INTERRUPT_: specifies the SDIO interrupt sources to be disabled. This parameter can be one or a combination of the following values:
 - SDIO_IT_CCRCFAIL: Command response received (CRC check failed) interrupt
 - SDIO_IT_DCRCFAIL: Data block sent/received (CRC check failed) interrupt
 - SDIO_IT_CTIMEOUT: Command response timeout interrupt
 - SDIO_IT_DTIMEOUT: Data timeout interrupt
 - SDIO_IT_TXUNDERR: Transmit FIFO underrun error interrupt
 - SDIO_IT_RXOVERR: Received FIFO overrun error interrupt
 - SDIO_IT_CMDREND: Command response received (CRC check passed) interrupt
 - SDIO_IT_CMDSENT: Command sent (no response required) interrupt
 - SDIO_IT_DATAEND: Data end (data counter, SDIDCOUNT, is zero) interrupt
 - SDIO_IT_STBITERR: Start bit not detected on all data signals in wide bus mode interrupt
 - SDIO_IT_DBCKEND: Data block sent/received (CRC check passed) interrupt
 - SDIO_IT_CMDACT: Command transfer in progress interrupt
 - SDIO_IT_TXACT: Data transmit in progress interrupt
 - SDIO_IT_RXACT: Data receive in progress interrupt

- interrupt
- SDIO_IT_TXFIFOHE: Transmit FIFO Half Empty interrupt
- SDIO_IT_RXFIFOHF: Receive FIFO Half Full interrupt
- SDIO_IT_TXFIFOF: Transmit FIFO full interrupt
- SDIO_IT_RXFIFOF: Receive FIFO full interrupt
- SDIO_IT_TXFIFOE: Transmit FIFO empty interrupt
- SDIO_IT_RXFIFOE: Receive FIFO empty interrupt
- SDIO_IT_TXDAVL: Data available in transmit FIFO interrupt
- SDIO_IT_RXDAVL: Data available in receive FIFO interrupt
- SDIO_IT_SDIOIT: SD I/O interrupt received interrupt
- SDIO_IT_CEATAEND: CE-ATA command completion signal received for CMD61 interrupt

Return value:

- None

__HAL_SD_SDIO_GET_FLAG**Description:**

- Check whether the specified SD flag is set or not.

Parameters:

- __HANDLE__: SD Handle
- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
 - SDIO_FLAG_CCRCFAIL: Command response received (CRC check failed)
 - SDIO_FLAG_DCRCFAIL: Data block sent/received (CRC check failed)
 - SDIO_FLAG_CTIMEOUT: Command response timeout
 - SDIO_FLAG_DTIMEOUT: Data timeout
 - SDIO_FLAG_TXUNDERR: Transmit FIFO underrun error
 - SDIO_FLAG_RXOVERR: Received FIFO overrun error
 - SDIO_FLAG_CMDREND: Command response received (CRC check passed)
 - SDIO_FLAG_CMDSENT: Command sent (no response required)
 - SDIO_FLAG_DATAEND: Data end (data counter, SDIDCOUNT, is zero)
 - SDIO_FLAG_STBITERR: Start bit not detected on all data signals in wide bus mode.



- SDIO_FLAG_DBCKEND: Data block sent/received (CRC check passed)
- SDIO_FLAG_CMDACT: Command transfer in progress
- SDIO_FLAG_TXACT: Data transmit in progress
- SDIO_FLAG_RXACT: Data receive in progress
- SDIO_FLAG_TXFIFOHE: Transmit FIFO Half Empty
- SDIO_FLAG_RXFIFOHF: Receive FIFO Half Full
- SDIO_FLAG_TXFIFOF: Transmit FIFO full
- SDIO_FLAG_RXFIFOF: Receive FIFO full
- SDIO_FLAG_TXFIFOE: Transmit FIFO empty
- SDIO_FLAG_RXFIFOE: Receive FIFO empty
- SDIO_FLAG_TXDAVL: Data available in transmit FIFO
- SDIO_FLAG_RXDAVL: Data available in receive FIFO
- SDIO_FLAG_SDIOIT: SD I/O interrupt received
- SDIO_FLAG_CEATAEND: CE-ATA command completion signal received for CMD61

Return value:

- The: new state of SD FLAG (SET or RESET).

_HAL_SD_SDIO_CLEAR_FLAG**Description:**

- Clear the SD's pending flags.

Parameters:

- _HANDLE_: SD Handle
- _FLAG_: specifies the flag to clear. This parameter can be one or a combination of the following values:
 - SDIO_FLAG_CCRCFAIL: Command response received (CRC check failed)
 - SDIO_FLAG_DCRCFAIL: Data block sent/received (CRC check failed)
 - SDIO_FLAG_CTIMEOUT: Command response timeout
 - SDIO_FLAG_DTIMEOUT: Data timeout
 - SDIO_FLAG_TXUNDERR: Transmit FIFO underrun error
 - SDIO_FLAG_RXOVERR: Received FIFO overrun error
 - SDIO_FLAG_CMDREND: Command response received (CRC check passed)
 - SDIO_FLAG_CMDSENT: Command sent (no response required)

- SDIO_FLAG_DATAEND: Data end (data counter, SDIDCOUNT, is zero)
- SDIO_FLAG_STBITERR: Start bit not detected on all data signals in wide bus mode
- SDIO_FLAG_DBCKEND: Data block sent/received (CRC check passed)
- SDIO_FLAG_SDIOIT: SD I/O interrupt received
- SDIO_FLAG_CEATAEND: CE-ATA command completion signal received for CMD61

Return value:

- None

_HAL_SD_SDIO_GET_IT**Description:**

- Check whether the specified SD interrupt has occurred or not.

Parameters:

- _HANDLE_: SD Handle
- _INTERRUPT_: specifies the SDIO interrupt source to check. This parameter can be one of the following values:
 - SDIO_IT_CCRCFAIL: Command response received (CRC check failed) interrupt
 - SDIO_IT_DCRCFAIL: Data block sent/received (CRC check failed) interrupt
 - SDIO_IT_CTIMEOUT: Command response timeout interrupt
 - SDIO_IT_DTIMEOUT: Data timeout interrupt
 - SDIO_IT_TXUNDERR: Transmit FIFO underrun error interrupt
 - SDIO_IT_RXOVERR: Received FIFO overrun error interrupt
 - SDIO_IT_CMDREND: Command response received (CRC check passed) interrupt
 - SDIO_IT_CMDSENT: Command sent (no response required) interrupt
 - SDIO_IT_DATAEND: Data end (data counter, SDIDCOUNT, is zero) interrupt
 - SDIO_IT_STBITERR: Start bit not detected on all data signals in wide bus mode interrupt
 - SDIO_IT_DBCKEND: Data block sent/received (CRC check passed) interrupt
 - SDIO_IT_CMDACT: Command transfer in progress interrupt
 - SDIO_IT_TXACT: Data transmit in progress interrupt
 - SDIO_IT_RXACT: Data receive in progress interrupt

- SDIO_IT_TXFIFOHE: Transmit FIFO Half Empty interrupt
- SDIO_IT_RXFIFOHF: Receive FIFO Half Full interrupt
- SDIO_IT_TXFIFOF: Transmit FIFO full interrupt
- SDIO_IT_RXFIFOF: Receive FIFO full interrupt
- SDIO_IT_TXFIFOE: Transmit FIFO empty interrupt
- SDIO_IT_RXFIFOE: Receive FIFO empty interrupt
- SDIO_IT_TXDAVL: Data available in transmit FIFO interrupt
- SDIO_IT_RXDAVL: Data available in receive FIFO interrupt
- SDIO_IT_SDIOIT: SD I/O interrupt received interrupt
- SDIO_IT_CEATAEND: CE-ATA command completion signal received for CMD61 interrupt

Return value:

- The: new state of SD IT (SET or RESET).

_HAL_SD_SDIO_CLEAR_IT**Description:**

- Clear the SD's interrupt pending bits.

Parameters:

- __HANDLE__: : SD Handle
- __INTERRUPT__: specifies the interrupt pending bit to clear. This parameter can be one or a combination of the following values:
 - SDIO_IT_CCRCFAIL: Command response received (CRC check failed) interrupt
 - SDIO_IT_DCRCFAIL: Data block sent/received (CRC check failed) interrupt
 - SDIO_IT_CTIMEOUT: Command response timeout interrupt
 - SDIO_IT_DTIMEOUT: Data timeout interrupt
 - SDIO_IT_TXUNDERR: Transmit FIFO underrun error interrupt
 - SDIO_IT_RXOVERR: Received FIFO overrun error interrupt
 - SDIO_IT_CMDREND: Command response received (CRC check passed) interrupt
 - SDIO_IT_CMDSENT: Command sent (no response required) interrupt
 - SDIO_IT_DATAEND: Data end (data counter, SDIO_DCOUNT, is zero) interrupt
 - SDIO_IT_STBITERR: Start bit not detected on all data signals in wide bus mode interrupt

- SDIO_IT_SDIOIT: SD I/O interrupt received interrupt
- SDIO_IT_CEATAEND: CE-ATA command completion signal received for CMD61

Return value:

- None

SD Handle Structure definition

[SD_InitTypeDef](#)

[SD_TypeDef](#)

39 HAL SMARTCARD Generic Driver

39.1 SMARTCARD Firmware driver registers structures

39.1.1 SMARTCARD_InitTypeDef

Data Fields

- *uint32_t BaudRate*
- *uint32_t WordLength*
- *uint32_t StopBits*
- *uint32_t Parity*
- *uint32_t Mode*
- *uint32_t CLKPolarity*
- *uint32_t CLKPhase*
- *uint32_t CLKLastBit*
- *uint32_t Prescaler*
- *uint32_t GuardTime*
- *uint32_t NACKState*

Field Documentation

- ***uint32_t SMARTCARD_InitTypeDef::BaudRate***
This member configures the SmartCard communication baud rate. The baud rate is computed using the following formula:

$$\text{IntegerDivider} = ((\text{PCLKx}) / (16 * (\text{hsmcard}\rightarrow\text{Init.BaudRate})))$$

$$\text{FractionalDivider} = ((\text{IntegerDivider} - ((\text{uint32_t}) \text{ IntegerDivider})) * 16) + 0.5$$
- ***uint32_t SMARTCARD_InitTypeDef::WordLength***
Specifies the number of data bits transmitted or received in a frame. This parameter can be a value of [**SMARTCARD_Word_Length**](#)
- ***uint32_t SMARTCARD_InitTypeDef::StopBits***
Specifies the number of stop bits transmitted. This parameter can be a value of [**SMARTCARD_Stop_Bits**](#)
- ***uint32_t SMARTCARD_InitTypeDef::Parity***
Specifies the parity mode. This parameter can be a value of [**SMARTCARD_Parity**](#)
Note:When parity is enabled, the computed parity is inserted at the MSB position of the transmitted data (9th bit when the word length is set to 9 data bits; 8th bit when the word length is set to 8 data bits).
- ***uint32_t SMARTCARD_InitTypeDef::Mode***
Specifies whether the Receive or Transmit mode is enabled or disabled. This parameter can be a value of [**SMARTCARD_Mode**](#)
- ***uint32_t SMARTCARD_InitTypeDef::CLKPolarity***
Specifies the steady state of the serial clock. This parameter can be a value of [**SMARTCARD_Clock_Polarity**](#)
- ***uint32_t SMARTCARD_InitTypeDef::CLKPhase***
Specifies the clock transition on which the bit capture is made. This parameter can be a value of [**SMARTCARD_Clock_Phase**](#)
- ***uint32_t SMARTCARD_InitTypeDef::CLKLastBit***
Specifies whether the clock pulse corresponding to the last transmitted data bit (MSB) has to be output on the SCLK pin in synchronous mode. This parameter can be a value of [**SMARTCARD_Last_Bit**](#)
- ***uint32_t SMARTCARD_InitTypeDef::Prescaler***
Specifies the SmartCard Prescaler value used for dividing the system clock to provide

the smartcard clock. The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency; This parameter can be a value of **SMARTCARD_Prescaler**

- **`uint32_t SMARTCARD_InitTypeDef::GuardTime`**
Specifies the SmartCard Guard Time value in terms of number of baud clocks
- **`uint32_t SMARTCARD_InitTypeDef::NACKState`**
Specifies the SmartCard NACK Transmission state This parameter can be a value of **SMARTCARD_NACK_State**

39.1.2 SMARTCARD_HandleTypeDef

Data Fields

- **`USART_TypeDef * Instance`**
- **`SMARTCARD_InitTypeDef Init`**
- **`uint8_t * pTxBuffPtr`**
- **`uint16_t TxXferSize`**
- **`uint16_t TxXferCount`**
- **`uint8_t * pRxBuffPtr`**
- **`uint16_t RxXferSize`**
- **`uint16_t RxXferCount`**
- **`DMA_HandleTypeDef * hdmatx`**
- **`DMA_HandleTypeDef * hdmarx`**
- **`HAL_LockTypeDef Lock`**
- **`__IO HAL_SMARTCARD_StateTypeDef State`**
- **`__IO uint32_t ErrorCode`**

Field Documentation

- **`USART_TypeDef* SMARTCARD_HandleTypeDef::Instance`**
USART registers base address
- **`SMARTCARD_InitTypeDef SMARTCARD_HandleTypeDef::Init`**
SmartCard communication parameters
- **`uint8_t* SMARTCARD_HandleTypeDef::pTxBuffPtr`**
Pointer to SmartCard Tx transfer Buffer
- **`uint16_t SMARTCARD_HandleTypeDef::TxXferSize`**
SmartCard Tx Transfer size
- **`uint16_t SMARTCARD_HandleTypeDef::TxXferCount`**
SmartCard Tx Transfer Counter
- **`uint8_t* SMARTCARD_HandleTypeDef::pRxBuffPtr`**
Pointer to SmartCard Rx transfer Buffer
- **`uint16_t SMARTCARD_HandleTypeDef::RxXferSize`**
SmartCard Rx Transfer size
- **`uint16_t SMARTCARD_HandleTypeDef::RxXferCount`**
SmartCard Rx Transfer Counter
- **`DMA_HandleTypeDef* SMARTCARD_HandleTypeDef::hdmatx`**
SmartCard Tx DMA Handle parameters
- **`DMA_HandleTypeDef* SMARTCARD_HandleTypeDef::hdmarx`**
SmartCard Rx DMA Handle parameters
- **`HAL_LockTypeDef SMARTCARD_HandleTypeDef::Lock`**
Locking object
- **`__IO HAL_SMARTCARD_StateTypeDef SMARTCARD_HandleTypeDef::State`**
SmartCard communication state
- **`__IO uint32_t SMARTCARD_HandleTypeDef::ErrorCode`**
SmartCard Error code

39.2 SMARTCARD Firmware driver API description

39.2.1 How to use this driver

The SMARTCARD HAL driver can be used as follows:

1. Declare a SMARTCARD_HandleTypeDef handle structure.
2. Initialize the SMARTCARD low level resources by implementing the HAL_SMARTCARD_MspInit() API:
 - a. Enable the interface clock of the USARTx associated to the SMARTCARD.
 - b. SMARTCARD pins configuration:
 - Enable the clock for the SMARTCARD GPIOs.
 - Configure the SMARTCARD pins as alternate function pull-up.
 - c. NVIC configuration if you need to use interrupt process (HAL_SMARTCARD_Transmit_IT() and HAL_SMARTCARD_Receive_IT() APIs):
 - Configure the USARTx interrupt priority.
 - Enable the NVIC USART IRQ handle.
 - d. DMA Configuration if you need to use DMA process (HAL_SMARTCARD_Transmit_DMA() and HAL_SMARTCARD_Receive_DMA() APIs):
 - Declare a DMA handle structure for the Tx/Rx channel.
 - Enable the DMAx interface clock.
 - Configure the declared DMA handle structure with the required Tx/Rx parameters.
 - Configure the DMA Tx/Rx channel.
 - Associate the initialized DMA handle to the SMARTCARD DMA Tx/Rx handle.
 - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx/Rx channel.
 - Configure the USARTx interrupt priority and enable the NVIC USART IRQ handle (used for last byte sending completion detection in DMA non circular mode)
3. Program the Baud Rate, Word Length , Stop Bit, Parity, Hardware flow control and Mode(Receiver/Transmitter) in the SMARTCARD Init structure.
4. Initialize the SMARTCARD registers by calling the HAL_SMARTCARD_Init() API:
 - This API configures also the low level Hardware GPIO, CLOCK, CORTEX...etc by calling the customized HAL_SMARTCARD_MspInit(&hsc) API. The specific SMARTCARD interrupts (Transmission complete interrupt, RXNE interrupt and Error Interrupts) will be managed using the macros __HAL_SMARTCARD_ENABLE_IT() and __HAL_SMARTCARD_DISABLE_IT() inside the transmit and receive process.
5. Three operation modes are available within this driver :

Polling mode IO operation

- Send an amount of data in blocking mode using HAL_SMARTCARD_Transmit()
- Receive an amount of data in blocking mode using HAL_SMARTCARD_Receive()

Interrupt mode IO operation

- Send an amount of data in non blocking mode using HAL_SMARTCARD_Transmit_IT()
- At transmission end of transfer HAL_SMARTCARD_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_SMARTCARD_TxCpltCallback

- Receive an amount of data in non blocking mode using HAL_SMARTCARD_Receive_IT()
- At reception end of transfer HAL_SMARTCARD_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_SMARTCARD_RxCpltCallback
- In case of transfer Error, HAL_SMARTCARD_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_SMARTCARD_ErrorCallback

DMA mode IO operation

- Send an amount of data in non blocking mode (DMA) using HAL_SMARTCARD_Transmit_DMA()
- At transmission end of transfer HAL_SMARTCARD_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_SMARTCARD_TxCpltCallback
- Receive an amount of data in non blocking mode (DMA) using HAL_SMARTCARD_Receive_DMA()
- At reception end of transfer HAL_SMARTCARD_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_SMARTCARD_RxCpltCallback
- In case of transfer Error, HAL_SMARTCARD_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_SMARTCARD_ErrorCallback

SMARTCARD HAL driver macros list

Below the list of most used macros in SMARTCARD HAL driver.

- __HAL_SMARTCARD_ENABLE: Enable the SMARTCARD peripheral
- __HAL_SMARTCARD_DISABLE: Disable the SMARTCARD peripheral
- __HAL_SMARTCARD_GET_FLAG : Check whether the specified SMARTCARD flag is set or not
- __HAL_SMARTCARD_CLEAR_FLAG : Clear the specified SMARTCARD pending flag
- __HAL_SMARTCARD_ENABLE_IT: Enable the specified SMARTCARD interrupt
- __HAL_SMARTCARD_DISABLE_IT: Disable the specified SMARTCARD interrupt
- __HAL_SMARTCARD_GET_IT_SOURCE: Check whether the specified SMARTCARD interrupt has occurred or not



You can refer to the SMARTCARD HAL driver header file for more useful macros

39.2.2 Initialization and Configuration functions

This subsection provides a set of functions allowing to initialize the USART in Smartcard mode.

The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard.

The USART can provide a clock to the smartcard through the SCLK output. In smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler.

- For the Smartcard mode only these parameters can be configured:
 - Baud Rate
 - Word Length => Should be 9 bits (8 bits + parity)
 - Stop Bit
 - Parity: => Should be enabled
 - USART polarity
 - USART phase
 - USART LastBit
 - Receiver/transmitter modes
 - Prescaler
 - GuardTime
 - NACKState: The Smartcard NACK state
- Recommended SmartCard interface configuration to get the Answer to Reset from the Card:
 - Word Length = 9 Bits
 - 1.5 Stop Bit
 - Even parity
 - BaudRate = 12096 baud
 - Tx and Rx enabled



It is also possible to choose 0.5 stop bit for receiving but it is recommended to use 1.5 stop bits for both transmitting and receiving to avoid switching between the two configurations.

The HAL_SMARTCARD_Init() function follows the USART SmartCard configuration procedure (details for the procedure are available in reference manual (RM0038)).

This section contains the following APIs:

- [**HAL_SMARTCARD_Init\(\)**](#)
- [**HAL_SMARTCARD_DelInit\(\)**](#)
- [**HAL_SMARTCARD_MspInit\(\)**](#)
- [**HAL_SMARTCARD_MspDelInit\(\)**](#)

39.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the SMARTCARD data transfers.

1. Smartcard is a single wire half duplex communication protocol. The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard.
2. The USART should be configured as:
 - 8 bits plus parity: where M=1 and PCE=1 in the USART_CR1 register
 - 1.5 stop bits when transmitting and receiving: where STOP=11 in the USART_CR2 register.
3. There are two modes of transfer:
 - Blocking mode: The communication is performed in polling mode. The HAL status of all data processing is returned by the same function after finishing transfer.
 - No-Blocking mode: The communication is performed using Interrupts or DMA, the relevant API's return the HAL status. The end of the data processing will be indicated through the dedicated SMARTCARD IRQ when using Interrupt mode or the DMA IRQ when using DMA mode. The HAL_SMARTCARD_TxCpltCallback(), HAL_SMARTCARD_RxCpltCallback()

user callbacks will be executed respectively at the end of the Transmit or Receive process. The HAL_SMARTCARD_ErrorCallback() user callback will be executed when a communication error is detected.

4. Blocking mode APIs are :
 - HAL_SMARTCARD_Transmit()
 - HAL_SMARTCARD_Receive()
5. Non Blocking mode APIs with Interrupt are :
 - HAL_SMARTCARD_Transmit_IT()
 - HAL_SMARTCARD_Receive_IT()
 - HAL_SMARTCARD_IRQHandler()
6. Non Blocking mode functions with DMA are :
 - HAL_SMARTCARD_Transmit_DMA()
 - HAL_SMARTCARD_Receive_DMA()
7. A set of Transfer Complete Callbacks are provided in non Blocking mode:
 - HAL_SMARTCARD_TxCpltCallback()
 - HAL_SMARTCARD_RxCpltCallback()
 - HAL_SMARTCARD_ErrorCallback()

This section contains the following APIs:

- [**HAL_SMARTCARD_Transmit\(\)**](#)
- [**HAL_SMARTCARD_Receive\(\)**](#)
- [**HAL_SMARTCARD_Transmit_IT\(\)**](#)
- [**HAL_SMARTCARD_Receive_IT\(\)**](#)
- [**HAL_SMARTCARD_Transmit_DMA\(\)**](#)
- [**HAL_SMARTCARD_Receive_DMA\(\)**](#)
- [**HAL_SMARTCARD_IRQHandler\(\)**](#)
- [**HAL_SMARTCARD_TxCpltCallback\(\)**](#)
- [**HAL_SMARTCARD_RxCpltCallback\(\)**](#)
- [**HAL_SMARTCARD_ErrorCallback\(\)**](#)

39.2.4 Peripheral State and Errors functions

This subsection provides a set of functions allowing to return the State of SmartCard communication process and also return Peripheral Errors occurred during communication process

- HAL_SMARTCARD_GetState() API can be helpful to check in run-time the state of the SMARTCARD peripheral.
- HAL_SMARTCARD_GetError() check in run-time errors that could be occurred during communication.

This section contains the following APIs:

- [**HAL_SMARTCARD_GetState\(\)**](#)
- [**HAL_SMARTCARD_GetError\(\)**](#)

39.2.5 Detailed description of functions

HAL_SMARTCARD_Init

Function name **HAL_StatusTypeDef HAL_SMARTCARD_Init
(SMARTCARD_HandleTypeDef * hsc)**

Function description Initializes the SmartCard mode according to the specified parameters in the SMARTCARD_HandleTypeDef and create the associated handle.

Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SMARTCARD_DelInit

Function name	HAL_StatusTypeDef HAL_SMARTCARD_DelInit (SMARTCARD_HandleTypeDef * hsc)
Function description	Deinitializes the SMARTCARD peripheral.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SMARTCARD_MspInit

Function name	void HAL_SMARTCARD_MspInit (SMARTCARD_HandleTypeDef * hsc)
Function description	SMARTCARD MSP Init.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SMARTCARD_MspDelInit

Function name	void HAL_SMARTCARD_MspDelInit (SMARTCARD_HandleTypeDef * hsc)
Function description	SMARTCARD MSP DelInit.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SMARTCARD_Transmit

Function name	HAL_StatusTypeDef HAL_SMARTCARD_Transmit (SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Sends an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module. • pData: Pointer to data buffer • Size: Amount of data to be sent • Timeout: Specify timeout value

Return values	<ul style="list-style-type: none"> • HAL: status
HAL_SMARTCARD_Receive	
Function name	HAL_StatusTypeDef HAL_SMARTCARD_Receive (SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Receive an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module. • pData: Pointer to data buffer • Size: Amount of data to be received • Timeout: Specify timeout value
Return values	<ul style="list-style-type: none"> • HAL: status
HAL_SMARTCARD_Transmit_IT	
Function name	HAL_StatusTypeDef HAL_SMARTCARD_Transmit_IT (SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t Size)
Function description	Sends an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module. • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status
HAL_SMARTCARD_Receive_IT	
Function name	HAL_StatusTypeDef HAL_SMARTCARD_Receive_IT (SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t Size)
Function description	Receives an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module. • pData: Pointer to data buffer • Size: Amount of data to be received
Return values	<ul style="list-style-type: none"> • HAL: status
HAL_SMARTCARD_Transmit_DMA	
Function name	HAL_StatusTypeDef HAL_SMARTCARD_Transmit_DMA (SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t Size)
Function description	Sends an amount of data in non-blocking mode.

Parameters	<ul style="list-style-type: none"> hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module. pData: Pointer to data buffer Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> HAL: status

HAL_SMARTCARD_Receive_DMA

Function name	HAL_StatusTypeDef HAL_SMARTCARD_Receive_DMA(SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t Size)
Function description	Receive an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module. pData: Pointer to data buffer Size: Amount of data to be received
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> When the SMARTCARD parity is enabled (PCE = 1) the data received contain the parity bit.

HAL_SMARTCARD_IRQHandler

Function name	void HAL_SMARTCARD_IRQHandler(SMARTCARD_HandleTypeDef * hsc)
Function description	This function handles SMARTCARD interrupt request.
Parameters	<ul style="list-style-type: none"> hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> None:

HAL_SMARTCARD_TxCpltCallback

Function name	void HAL_SMARTCARD_TxCpltCallback(SMARTCARD_HandleTypeDef * hsc)
Function description	Tx Transfer completed callback.
Parameters	<ul style="list-style-type: none"> hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> None:

HAL_SMARTCARD_RxCpltCallback

Function name	void HAL_SMARTCARD_RxCpltCallback(SMARTCARD_HandleTypeDef * hsc)
Function description	Rx Transfer completed callback.

Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SMARTCARD_ErrorCallback

Function name	void HAL_SMARTCARD_ErrorCallback(SMARTCARD_HandleTypeDef * hsc)
Function description	SMARTCARD error callback.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SMARTCARD_GetState

Function name	HAL_SMARTCARD_StateTypeDef HAL_SMARTCARD_GetState(SMARTCARD_HandleTypeDef * hsc)
Function description	Returns the SMARTCARD state.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_SMARTCARD_GetError

Function name	uint32_t HAL_SMARTCARD_GetError(SMARTCARD_HandleTypeDef * hsc)
Function description	Return the SMARTCARD error code.
Parameters	<ul style="list-style-type: none"> • hsc: Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.
Return values	<ul style="list-style-type: none"> • SMARTCARD: Error Code

39.3 SMARTCARD Firmware driver defines

39.3.1 SMARTCARD

SMARTCARD Clock Phase

SMARTCARD_PHASE_1EDGE

SMARTCARD_PHASE_2EDGE

SMARTCARD Clock Polarity

SMARTCARD_POLARITY_LOW

SMARTCARD_POLARITY_HIGH

SMARTCARD DMA requests`SMARTCARD_DMAREQ_TX``SMARTCARD_DMAREQ_RX`***SMARTCARD Error Codes***`HAL_SMARTCARD_ERROR_NONE` No error`HAL_SMARTCARD_ERROR_PE` Parity error`HAL_SMARTCARD_ERROR_NE` Noise error`HAL_SMARTCARD_ERROR_FE` frame error`HAL_SMARTCARD_ERROR_ORE` Overrun error`HAL_SMARTCARD_ERROR_DMA` DMA transfer error***SMARTCARD Exported Macros***

`__HAL_SMARTCARD_RESET_HANDLE_STA` **Description:**

`TE`

- Reset SMARTCARD handle state.

Parameters:

- `__HANDLE__`: specifies the SMARTCARD Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_FLUSH_DRREGISTER` **Description:**

- Flush the Smartcard DR register.

Parameters:

- `__HANDLE__`: specifies the SMARTCARD Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_GET_FLAG` **Description:**

- Check whether the specified Smartcard flag is set or not.

Parameters:

- `__HANDLE__`: specifies the SMARTCARD Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending

on device).

- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
 - SMARTCARD_FLAG_TXE: Transmit data register empty flag
 - SMARTCARD_FLAG_TC: Transmission Complete flag
 - SMARTCARD_FLAG_RXNE: Receive data register not empty flag
 - SMARTCARD_FLAG_IDLE: Idle Line detection flag
 - SMARTCARD_FLAG_ORE: OverRun Error flag
 - SMARTCARD_FLAG_NE: Noise Error flag
 - SMARTCARD_FLAG_FE: Framing Error flag
 - SMARTCARD_FLAG_PE: Parity Error flag

Return value:

- The new state of __FLAG__ (TRUE or FALSE).

_HAL_SMARTCARD_CLEAR_FLAG

Description:

- Clear the specified Smartcard pending flags.

Parameters:

- __HANDLE__: specifies the SMARTCARD Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).
- __FLAG__: specifies the flag to check. This parameter can be any combination of the following values:
 - SMARTCARD_FLAG_TC: Transmission Complete flag.
 - SMARTCARD_FLAG_RXNE: Receive data register not empty flag.

Return value:

- None
- None

Notes:

- PE (Parity error), FE (Framing

error), NE (Noise error) and ORE (OverRun error) flags are cleared by software sequence: a read operation to USART_SR register followed by a read operation to USART_DR register. RXNE flag can be also cleared by a read to the USART_DR register. TC flag can be also cleared by software sequence: a read operation to USART_SR register followed by a write operation to USART_DR register. TXE flag is cleared only by a write to the USART_DR register.

`__HAL_SMARTCARD_CLEAR_PEFLAG`

Description:

- Clear the SMARTCARD PE pending flag.

Parameters:

- `__HANDLE__`: specifies the USART Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_CLEAR_FEFLAG`

Description:

- Clear the SMARTCARD FE pending flag.

Parameters:

- `__HANDLE__`: specifies the USART Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_CLEAR_NEFLAG`

Description:

- Clear the SMARTCARD NE pending flag.

Parameters:

- `__HANDLE__`: specifies the USART Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_CLEAR_OREFLAG`**Description:**

- Clear the SMARTCARD ORE pending flag.

Parameters:

- `__HANDLE__`: specifies the USART Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_CLEAR_IDLEFLAG`**Description:**

- Clear the SMARTCARD IDLE pending flag.

Parameters:

- `__HANDLE__`: specifies the USART Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_ENABLE_IT`**Description:**

- Enable the specified SmartCard interrupt.

Parameters:

- `__HANDLE__`: specifies the SMARTCARD Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).
- `__INTERRUPT__`: specifies the SMARTCARD interrupt to enable. This parameter can be one of the following values:
 - `SMARTCARD_IT_TXE`: Transmit Data Register empty interrupt
 - `SMARTCARD_IT_TC`: Transmission complete interrupt
 - `SMARTCARD_IT_RXNE`: Receive Data register not

- empty interrupt
- SMARTCARD_IT_IDLE: Idle line detection interrupt
- SMARTCARD_IT_PE: Parity Error interrupt
- SMARTCARD_IT_ERR: Error interrupt(Frame error, noise error, overrun error)

Return value:

- None

_HAL_SMARTCARD_DISABLE_IT

- Disable the specified SmartCard interrupts.

Parameters:

- HANDLE: specifies the SMARTCARD Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).
- INTERRUPT: specifies the SMARTCARD interrupt to disable. This parameter can be one of the following values:
 - SMARTCARD_IT_TXE: Transmit Data Register empty interrupt
 - SMARTCARD_IT_TC: Transmission complete interrupt
 - SMARTCARD_IT_RXNE: Receive Data register not empty interrupt
 - SMARTCARD_IT_IDLE: Idle line detection interrupt
 - SMARTCARD_IT_PE: Parity Error interrupt
 - SMARTCARD_IT_ERR: Error interrupt(Frame error, noise error, overrun error)

_HAL_SMARTCARD_GET_IT_SOURCE**Description:**

- Check whether the specified SmartCard interrupt has occurred or not.

Parameters:

- HANDLE: specifies the SMARTCARD Handle. SMARTCARD Handle selects the USARTx peripheral (USART

availability and x value depending on device).

- __IT__: specifies the SMARTCARD interrupt source to check. This parameter can be one of the following values:
 - SMARTCARD_IT_TXE: Transmit Data Register empty interrupt
 - SMARTCARD_IT_TC: Transmission complete interrupt
 - SMARTCARD_IT_RXNE: Receive Data register not empty interrupt
 - SMARTCARD_IT_IDLE: Idle line detection interrupt
 - SMARTCARD_IT_ERR: Error interrupt
 - SMARTCARD_IT_PE: Parity Error interrupt

Return value:

- The new state of __IT__ (TRUE or FALSE).

Description:

- Enables the SMARTCARD one bit sample method.

Parameters:

- __HANDLE__: specifies the SMARTCARD Handle.

Return value:

- None

Description:

- Disables the SMARTCARD one bit sample method.

Parameters:

- __HANDLE__: specifies the SMARTCARD Handle.

Return value:

- None

Description:

- Enable the USART associated to the SMARTCARD Handle.

Parameters:

- __HANDLE__: specifies the

SMARTCARD Handle.
SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_DISABLE`

Description:

- Disable the USART associated to the SMARTCARD Handle.

Parameters:

- `__HANDLE__`: specifies the SMARTCARD Handle.
SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_SMARTCARD_DMA_REQUEST_ENAB`
`LE`

Description:

- Enable the SmartCard DMA request.

Parameters:

- `__HANDLE__`: specifies the SmartCard Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).
- `__REQUEST__`: specifies the SmartCard DMA request. This parameter can be one of the following values:
 - `SMARTCARD_DMAREQ_TX`: SmartCard DMA transmit request
 - `SMARTCARD_DMAREQ_RX` : SmartCard DMA receive request

Return value:

- None

`__HAL_SMARTCARD_DMA_REQUEST_DISA`
`BLE`

Description:

- Disable the SmartCard DMA request.

Parameters:

- `__HANDLE__`: specifies the

SmartCard Handle. SMARTCARD Handle selects the USARTx peripheral (USART availability and x value depending on device).

- _REQUEST_: specifies the SmartCard DMA request. This parameter can be one of the following values:
 - SMARTCARD_DMAREQ_TX: SmartCard DMA transmit request
 - SMARTCARD_DMAREQ_RX : SmartCard DMA receive request

Return value:

- None

SMARTCARD Flags

SMARTCARD_FLAG_TXE
 SMARTCARD_FLAG_TC
 SMARTCARD_FLAG_RXNE
 SMARTCARD_FLAG_IDLE
 SMARTCARD_FLAG_ORE
 SMARTCARD_FLAG_NE
 SMARTCARD_FLAG_FE
 SMARTCARD_FLAG_PE

SMARTCARD Interrupts Definition

SMARTCARD_IT_PE
 SMARTCARD_IT_TXE
 SMARTCARD_IT_TC
 SMARTCARD_IT_RXNE
 SMARTCARD_IT_IDLE
 SMARTCARD_IT_ERR

SMARTCARD Last Bit

SMARTCARD_LASTBIT_DISABLE
 SMARTCARD_LASTBIT_ENABLE

SMARTCARD Mode

SMARTCARD_MODE_RX
 SMARTCARD_MODE_TX
 SMARTCARD_MODE_TX_RX

SMARTCARD NACK State

SMARTCARD_NACK_ENABLE

`SMARTCARD_NACK_DISABLE`

SMARTCARD One Bit Sampling Method

`SMARTCARD_ONE_BIT_SAMPLE_DISABLE`

`SMARTCARD_ONE_BIT_SAMPLE_ENABLE`

SMARTCARD Parity

`SMARTCARD_PARITY_EVEN`

`SMARTCARD_PARITY_ODD`

SMARTCARD Prescaler

<code>SMARTCARD_PRESCALER_SYSCLK_DIV2</code>	SYSCLK divided by 2
<code>SMARTCARD_PRESCALER_SYSCLK_DIV4</code>	SYSCLK divided by 4
<code>SMARTCARD_PRESCALER_SYSCLK_DIV6</code>	SYSCLK divided by 6
<code>SMARTCARD_PRESCALER_SYSCLK_DIV8</code>	SYSCLK divided by 8
<code>SMARTCARD_PRESCALER_SYSCLK_DIV10</code>	SYSCLK divided by 10
<code>SMARTCARD_PRESCALER_SYSCLK_DIV12</code>	SYSCLK divided by 12
<code>SMARTCARD_PRESCALER_SYSCLK_DIV14</code>	SYSCLK divided by 14
<code>SMARTCARD_PRESCALER_SYSCLK_DIV16</code>	SYSCLK divided by 16
<code>SMARTCARD_PRESCALER_SYSCLK_DIV18</code>	SYSCLK divided by 18
<code>SMARTCARD_PRESCALER_SYSCLK_DIV20</code>	SYSCLK divided by 20
<code>SMARTCARD_PRESCALER_SYSCLK_DIV22</code>	SYSCLK divided by 22
<code>SMARTCARD_PRESCALER_SYSCLK_DIV24</code>	SYSCLK divided by 24
<code>SMARTCARD_PRESCALER_SYSCLK_DIV26</code>	SYSCLK divided by 26
<code>SMARTCARD_PRESCALER_SYSCLK_DIV28</code>	SYSCLK divided by 28
<code>SMARTCARD_PRESCALER_SYSCLK_DIV30</code>	SYSCLK divided by 30
<code>SMARTCARD_PRESCALER_SYSCLK_DIV32</code>	SYSCLK divided by 32
<code>SMARTCARD_PRESCALER_SYSCLK_DIV34</code>	SYSCLK divided by 34
<code>SMARTCARD_PRESCALER_SYSCLK_DIV36</code>	SYSCLK divided by 36
<code>SMARTCARD_PRESCALER_SYSCLK_DIV38</code>	SYSCLK divided by 38
<code>SMARTCARD_PRESCALER_SYSCLK_DIV40</code>	SYSCLK divided by 40
<code>SMARTCARD_PRESCALER_SYSCLK_DIV42</code>	SYSCLK divided by 42
<code>SMARTCARD_PRESCALER_SYSCLK_DIV44</code>	SYSCLK divided by 44
<code>SMARTCARD_PRESCALER_SYSCLK_DIV46</code>	SYSCLK divided by 46
<code>SMARTCARD_PRESCALER_SYSCLK_DIV48</code>	SYSCLK divided by 48
<code>SMARTCARD_PRESCALER_SYSCLK_DIV50</code>	SYSCLK divided by 50
<code>SMARTCARD_PRESCALER_SYSCLK_DIV52</code>	SYSCLK divided by 52
<code>SMARTCARD_PRESCALER_SYSCLK_DIV54</code>	SYSCLK divided by 54
<code>SMARTCARD_PRESCALER_SYSCLK_DIV56</code>	SYSCLK divided by 56

SMARTCARD_PRESCALER_SYSCLK_DIV58 SYSCLK divided by 58

SMARTCARD_PRESCALER_SYSCLK_DIV60 SYSCLK divided by 60

SMARTCARD_PRESCALER_SYSCLK_DIV62 SYSCLK divided by 62

SMARTCARD Number of Stop Bits

SMARTCARD_STOPBITS_0_5

SMARTCARD_STOPBITS_1_5

SMARTCARD Word Length

SMARTCARD_WORDLENGTH_9B

40 HAL SPI Generic Driver

40.1 SPI Firmware driver registers structures

40.1.1 SPI_InitTypeDef

Data Fields

- *uint32_t Mode*
- *uint32_t Direction*
- *uint32_t DataSize*
- *uint32_t CLKPolarity*
- *uint32_t CLKPhase*
- *uint32_t NSS*
- *uint32_t BaudRatePrescaler*
- *uint32_t FirstBit*
- *uint32_t TIMode*
- *uint32_t CRCCalculation*
- *uint32_t CRCPolynomial*

Field Documentation

- ***uint32_t SPI_InitTypeDef::Mode***
Specifies the SPI operating mode. This parameter can be a value of [**SPI_mode**](#)
- ***uint32_t SPI_InitTypeDef::Direction***
Specifies the SPI Directional mode state. This parameter can be a value of [**SPI_Direction_mode**](#)
- ***uint32_t SPI_InitTypeDef::DataSize***
Specifies the SPI data size. This parameter can be a value of [**SPI_data_size**](#)
- ***uint32_t SPI_InitTypeDef::CLKPolarity***
Specifies the serial clock steady state. This parameter can be a value of [**SPI_Clock_Polarity**](#)
- ***uint32_t SPI_InitTypeDef::CLKPhase***
Specifies the clock active edge for the bit capture. This parameter can be a value of [**SPI_Clock_Phase**](#)
- ***uint32_t SPI_InitTypeDef::NSS***
Specifies whether the NSS signal is managed by hardware (NSS pin) or by software using the SSI bit. This parameter can be a value of [**SPI_Slave_Select_management**](#)
- ***uint32_t SPI_InitTypeDef::BaudRatePrescaler***
Specifies the Baud Rate prescaler value which will be used to configure the transmit and receive SCK clock. This parameter can be a value of [**SPI_BaudRate_Prescaler**](#)
Note:The communication clock is derived from the master clock. The slave clock does not need to be set
- ***uint32_t SPI_InitTypeDef::FirstBit***
Specifies whether data transfers start from MSB or LSB bit. This parameter can be a value of [**SPI_MSB_LSB_transmission**](#)
- ***uint32_t SPI_InitTypeDef::TIMode***
Specifies if the TI mode is enabled or not. This parameter can be a value of [**SPI_TI_mode**](#)
- ***uint32_t SPI_InitTypeDef::CRCCalculation***
Specifies if the CRC calculation is enabled or not. This parameter can be a value of [**SPI_CRC_Calculation**](#)

- ***uint32_t SPI_InitTypeDef::CRCPolynomial***
Specifies the polynomial used for the CRC calculation. This parameter must be a number between Min_Data = 0 and Max_Data = 65535

40.1.2 ***_SPI_HandleTypeDef***

Data Fields

- ***SPI_TypeDef * Instance***
- ***SPI_InitTypeDef Init***
- ***uint8_t * pTxBuffPtr***
- ***uint16_t TxXferSize***
- ***_IO uint16_t TxXferCount***
- ***uint8_t * pRxBuffPtr***
- ***uint16_t RxXferSize***
- ***_IO uint16_t RxXferCount***
- ***DMA_HandleTypeDef * hdmatx***
- ***DMA_HandleTypeDef * hdmarx***
- ***void(* RxISR***
- ***void(* TxISR***
- ***HAL_LockTypeDef Lock***
- ***_IO HAL_SPI_StateTypeDef State***
- ***_IO uint32_t ErrorCode***

Field Documentation

- ***SPI_TypeDef* __SPI_HandleTypeDef::Instance***
- ***SPI_InitTypeDef __SPI_HandleTypeDef::Init***
- ***uint8_t* __SPI_HandleTypeDef::pTxBuffPtr***
- ***uint16_t __SPI_HandleTypeDef::TxXferSize***
- ***_IO uint16_t __SPI_HandleTypeDef::TxXferCount***
- ***uint8_t* __SPI_HandleTypeDef::pRxBuffPtr***
- ***uint16_t __SPI_HandleTypeDef::RxXferSize***
- ***_IO uint16_t __SPI_HandleTypeDef::RxXferCount***
- ***DMA_HandleTypeDef* __SPI_HandleTypeDef::hdmatx***
- ***DMA_HandleTypeDef* __SPI_HandleTypeDef::hdmarx***
- ***void(* __SPI_HandleTypeDef::RxISR)(struct __SPI_HandleTypeDef *hspi)***
- ***void(* __SPI_HandleTypeDef::TxISR)(struct __SPI_HandleTypeDef *hspi)***
- ***HAL_LockTypeDef __SPI_HandleTypeDef::Lock***
- ***_IO HAL_SPI_StateTypeDef __SPI_HandleTypeDef::State***
- ***_IO uint32_t __SPI_HandleTypeDef::ErrorCode***

40.2 SPI Firmware driver API description

40.2.1 How to use this driver

The SPI HAL driver can be used as follows:

1. Declare a SPI_HandleTypeDef handle structure, for example: SPI_HandleTypeDef hspi;
2. Initialize the SPI low level resources by implementing the HAL_SPI_MspInit ()API:
 - a. Enable the SPIx interface clock
 - b. SPI pins configuration
 - Enable the clock for the SPI GPIOs
 - Configure these SPI pins as alternate function push-pull
 - c. NVIC configuration if you need to use interrupt process

- Configure the SPIx interrupt priority
- Enable the NVIC SPI IRQ handle
- d. DMA Configuration if you need to use DMA process
 - Declare a DMA_HandleTypeDef handle structure for the transmit or receive Channel
 - Enable the DMAx clock
 - Configure the DMA handle parameters
 - Configure the DMA Tx or Rx Channel
 - Associate the initialized hdma_tx(or _rx) handle to the hspi DMA Tx (or Rx) handle
 - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx or Rx Channel
- 3. Program the Mode, Direction , Data size, Baudrate Prescaler, NSS management, Clock polarity and phase, FirstBit and CRC configuration in the hspi Init structure.
- 4. Initialize the SPI registers by calling the HAL_SPI_Init() API:
 - This API configures also the low level Hardware GPIO, CLOCK, CORTEX...etc by calling the customized HAL_SPI_MspInit() API.

Circular mode restriction:

1. The DMA circular mode cannot be used when the SPI is configured in these modes:
 - a. Master 2Lines RxOnly
 - b. Master 1Line Rx
2. The CRC feature is not managed when the DMA circular mode is enabled
3. When the SPI DMA Pause/Stop features are used, we must use the following APIs the HAL_SPI_DMAPause()// HAL_SPI_DMAStop() only under the SPI callbacks

40.2.2 Initialization and de-initialization functions

This subsection provides a set of functions allowing to initialize and de-initialiaze the SPIx peripheral:

- User must implement HAL_SPI_MspInit() function in which he configures all related peripherals resources (CLOCK, GPIO, DMA, IT and NVIC).
- Call the function HAL_SPI_Init() to configure the selected device with the selected configuration:
 - Mode
 - Direction
 - Data Size
 - Clock Polarity and Phase
 - NSS Management
 - BaudRate Prescaler
 - FirstBit
 - TIMode
 - CRC Calculation
 - CRC Polynomial if CRC enabled
- Call the function HAL_SPI_DelInit() to restore the default configuration of the selected SPIx periperal.

This section contains the following APIs:

- [**HAL_SPI_Init\(\)**](#)
- [**HAL_SPI_DelInit\(\)**](#)
- [**HAL_SPI_MspInit\(\)**](#)
- [**HAL_SPI_MspDelInit\(\)**](#)

40.2.3 IO operation functions

The SPI supports master and slave mode :

1. There are two modes of transfer:
 - Blocking mode: The communication is performed in polling mode. The HAL status of all data processing is returned by the same function after finishing transfer.
 - No-Blocking mode: The communication is performed using Interrupts or DMA, These APIs return the HAL status. The end of the data processing will be indicated through the dedicated SPI IRQ when using Interrupt mode or the DMA IRQ when using DMA mode. The `HAL_SPI_TxCpltCallback()`, `HAL_SPI_RxCpltCallback()` and `HAL_SPI_TxRxCpltCallback()` user callbacks will be executed respectively at the end of the transmit or Receive process The `HAL_SPI_ErrorCallback()`user callback will be executed when a communication error is detected
2. APIs provided for these 2 transfer modes (Blocking mode or Non blocking mode using either Interrupt or DMA) exist for 1Line (simplex) and 2Lines (full duplex) modes.

This section contains the following APIs:

- `HAL_SPI_Transmit()`
- `HAL_SPI_Receive()`
- `HAL_SPI_TransmitReceive()`
- `HAL_SPI_Transmit_IT()`
- `HAL_SPI_Receive_IT()`
- `HAL_SPI_TransmitReceive_IT()`
- `HAL_SPI_Transmit_DMA()`
- `HAL_SPI_Receive_DMA()`
- `HAL_SPI_TransmitReceive_DMA()`
- `HAL_SPI_DMAPause()`
- `HAL_SPI_DMAResume()`
- `HAL_SPI_DMAStop()`
- `HAL_SPI_IRQHandler()`
- `HAL_SPI_TxCpltCallback()`
- `HAL_SPI_RxCpltCallback()`
- `HAL_SPI_TxRxCpltCallback()`
- `HAL_SPI_TxHalfCpltCallback()`
- `HAL_SPI_RxHalfCpltCallback()`
- `HAL_SPI_TxRxHalfCpltCallback()`
- `HAL_SPI_ErrorCallback()`

40.2.4 Peripheral State and Errors functions

This subsection provides a set of functions allowing to control the SPI.

- `HAL_SPI_GetState()` API can be helpful to check in run-time the state of the SPI peripheral
- `HAL_SPI_GetError()` check in run-time Errors occurring during communication

This section contains the following APIs:

- `HAL_SPI_GetState()`
- `HAL_SPI_GetError()`

40.2.5 Detailed description of functions

HAL_SPI_Init

Function name	HAL_StatusTypeDef HAL_SPI_Init (SPI_HandleTypeDef * hspi)
Function description	Initializes the SPI according to the specified parameters in the SPI_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SPI_DeInit

Function name	HAL_StatusTypeDef HAL_SPI_DeInit (SPI_HandleTypeDef * hspi)
Function description	DeInitializes the SPI peripheral.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SPI_MspInit

Function name	void HAL_SPI_MspInit (SPI_HandleTypeDef * hspi)
Function description	SPI MSP Init.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SPI_MspDeInit

Function name	void HAL_SPI_MspDeInit (SPI_HandleTypeDef * hspi)
Function description	SPI MSP DeInit.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SPI_Transmit

Function name	HAL_StatusTypeDef HAL_SPI_Transmit (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Transmit an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • pData: pointer to data buffer • Size: amount of data to be sent • Timeout: Timeout duration

Return values	<ul style="list-style-type: none"> HAL: status
HAL_SPI_Receive	
Function name	HAL_StatusTypeDef HAL_SPI_Receive (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Receive an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. pData: pointer to data buffer Size: amount of data to be sent Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status
HAL_SPI_TransmitReceive	
Function name	HAL_StatusTypeDef HAL_SPI_TransmitReceive (SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)
Function description	Transmit and Receive an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. pTxData: pointer to transmission data buffer pRxData: pointer to reception data buffer to be Size: amount of data to be sent Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status
HAL_SPI_Transmit_IT	
Function name	HAL_StatusTypeDef HAL_SPI_Transmit_IT (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)
Function description	Transmit an amount of data in no-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. pData: pointer to data buffer Size: amount of data to be sent
Return values	<ul style="list-style-type: none"> HAL: status
HAL_SPI_Receive_IT	
Function name	HAL_StatusTypeDef HAL_SPI_Receive_IT (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)
Function description	Receive an amount of data in no-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. pData: pointer to data buffer

- **Size:** amount of data to be sent
- Return values • **HAL:** status

HAL_SPI_TransmitReceive_IT

Function name	HAL_StatusTypeDef HAL_SPI_TransmitReceive_IT (SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)
Function description	Transmit and Receive an amount of data in no-blocking mode with Interrupt.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • pTxData: pointer to transmission data buffer • pRxData: pointer to reception data buffer to be • Size: amount of data to be sent
Return values	• HAL: status

HAL_SPI_Transmit_DMA

Function name	HAL_StatusTypeDef HAL_SPI_Transmit_DMA (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)
Function description	Transmit an amount of data in no-blocking mode with DMA.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • pData: pointer to data buffer • Size: amount of data to be sent
Return values	• HAL: status

HAL_SPI_Receive_DMA

Function name	HAL_StatusTypeDef HAL_SPI_Receive_DMA (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)
Function description	Receive an amount of data in no-blocking mode with DMA.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • pData: pointer to data buffer • Size: amount of data to be sent
Return values	• HAL: status

Notes

- When the CRC feature is enabled the pData Length must be Size + 1.

HAL_SPI_TransmitReceive_DMA

Function name	HAL_StatusTypeDef HAL_SPI_TransmitReceive_DMA (SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)
Function description	Transmit and Receive an amount of data in no-blocking mode with

	DMA.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • pTxData: pointer to transmission data buffer • pRxData: pointer to reception data buffer • Size: amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When the CRC feature is enabled the pRxData Length must be Size + 1

HAL_SPI_DMAPause

Function name	HAL_StatusTypeDef HAL_SPI_DMAPause (SPI_HandleTypeDef * hspi)
Function description	Pauses the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for the specified SPI module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SPI_DMAResume

Function name	HAL_StatusTypeDef HAL_SPI_DMAResume (SPI_HandleTypeDef * hspi)
Function description	Resumes the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for the specified SPI module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SPI_DMAStop

Function name	HAL_StatusTypeDef HAL_SPI_DMAStop (SPI_HandleTypeDef * hspi)
Function description	Stops the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SPI_IRQHandler

Function name	void HAL_SPI_IRQHandler (SPI_HandleTypeDef * hspi)
Function description	This function handles SPI interrupt request.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SPI_TxCpltCallback

Function name	void HAL_SPI_TxCpltCallback (SPI_HandleTypeDef * hspi)
Function description	Tx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none">• hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none">• None:

HAL_SPI_RxCpltCallback

Function name	void HAL_SPI_RxCpltCallback (SPI_HandleTypeDef * hspi)
Function description	Rx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none">• hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none">• None:

HAL_SPI_TxRxCpltCallback

Function name	void HAL_SPI_TxRxCpltCallback (SPI_HandleTypeDef * hspi)
Function description	Tx and Rx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none">• hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none">• None:

HAL_SPI_ErrorCallback

Function name	void HAL_SPI_ErrorCallback (SPI_HandleTypeDef * hspi)
Function description	SPI error callbacks.
Parameters	<ul style="list-style-type: none">• hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none">• None:

HAL_SPI_TxHalfCpltCallback

Function name	void HAL_SPI_TxHalfCpltCallback (SPI_HandleTypeDef * hspi)
Function description	Tx Half Transfer completed callbacks.
Parameters	<ul style="list-style-type: none">• hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none">• None:

HAL_SPI_RxHalfCpltCallback

Function name	void HAL_SPI_RxHalfCpltCallback (SPI_HandleTypeDef * hspi)
Function description	Rx Half Transfer completed callbacks.

Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SPI_TxRxHalfCpltCallback

Function name	void HAL_SPI_TxRxHalfCpltCallback (SPI_HandleTypeDef * hspi)
Function description	Tx and Rx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SPI_GetState

Function name	HAL_SPI_StateTypeDef HAL_SPI_GetState (SPI_HandleTypeDef * hspi)
Function description	Return the SPI state.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_SPI_GetError

Function name	uint32_t HAL_SPI_GetError (SPI_HandleTypeDef * hspi)
Function description	Return the SPI error code.
Parameters	<ul style="list-style-type: none"> • hspi: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	<ul style="list-style-type: none"> • SPI: Error Code

40.3 SPI Firmware driver defines

40.3.1 SPI

SPI BaudRate Prescaler

SPI_BAUDRATEPRESCALER_2
SPI_BAUDRATEPRESCALER_4
SPI_BAUDRATEPRESCALER_8
SPI_BAUDRATEPRESCALER_16
SPI_BAUDRATEPRESCALER_32
SPI_BAUDRATEPRESCALER_64
SPI_BAUDRATEPRESCALER_128
SPI_BAUDRATEPRESCALER_256
IS_SPI_BAUDRATE_PRESCALER

SPI Clock Phase

SPI_PHASE_1EDGE

SPI_PHASE_2EDGE

IS_SPI_CPHA

SPI Clock Polarity

SPI_POLARITY_LOW

SPI_POLARITY_HIGH

IS_SPI_CPOL

SPI CRC Calculation

SPI_CRCCALCULATION_DISABLE

SPI_CRCCALCULATION_ENABLE

IS_SPI_CRC_CALCULATION

IS_SPI_CRC_POLYNOMIAL

SPI data size

SPI_DATASIZE_8BIT

SPI_DATASIZE_16BIT

IS_SPI_DATASIZE

SPI Direction mode

SPI_DIRECTION_2LINES

SPI_DIRECTION_2LINES_RXONLY

SPI_DIRECTION_1LINE

IS_SPI_DIRECTION_MODE

IS_SPI_DIRECTION_2LINES_OR_1LINE

IS_SPI_DIRECTION_2LINES

SPI Error Codes

HAL_SPI_ERROR_NONE No error

HAL_SPI_ERROR_MODF MODF error

HAL_SPI_ERROR_CRC CRC error

HAL_SPI_ERROR_OVR OVR error

HAL_SPI_ERROR_FRE FRE error

HAL_SPI_ERROR_DMA DMA transfer error

HAL_SPI_ERROR_FLAG Flag: RXNE,TXE, BSY

SPI Exported Macros__HAL_SPI_RESET_HANDLE_STATE **Description:**

- Reset SPI handle state.

Parameters:

- __HANDLE__: specifies the SPI handle.

This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.

Return value:

- None

`__HAL_SPI_ENABLE_IT`

Description:

- Enable or disable the specified SPI interrupts.

Parameters:

- `__HANDLE__`: specifies the SPI handle. This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.
- `__INTERRUPT__`: specifies the interrupt source to enable or disable. This parameter can be one of the following values:
 - `SPI_IT_TXE`: Tx buffer empty interrupt enable
 - `SPI_IT_RXNE`: RX buffer not empty interrupt enable
 - `SPI_IT_ERR`: Error interrupt enable

Return value:

- None

`__HAL_SPI_DISABLE_IT`

`__HAL_SPI_GET_IT_SOURCE`

Description:

- Check if the specified SPI interrupt source is enabled or disabled.

Parameters:

- `__HANDLE__`: specifies the SPI handle. This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.
- `__INTERRUPT__`: specifies the SPI interrupt source to check. This parameter can be one of the following values:
 - `SPI_IT_TXE`: Tx buffer empty interrupt enable
 - `SPI_IT_RXNE`: RX buffer not empty interrupt enable
 - `SPI_IT_ERR`: Error interrupt enable

Return value:

- The: new state of `__IT__` (TRUE or FALSE).

`__HAL_SPI_GET_FLAG`

Description:

- Check whether the specified SPI flag is set or not.

Parameters:

- `__HANDLE__`: specifies the SPI handle. This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.

- 3 to select the SPI peripheral.
- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
 - SPI_FLAG_RXNE: Receive buffer not empty flag
 - SPI_FLAG_TXE: Transmit buffer empty flag
 - SPI_FLAG_CRCERR: CRC error flag
 - SPI_FLAG_MODF: Mode fault flag
 - SPI_FLAG_OVR: Overrun flag
 - SPI_FLAG_BSY: Busy flag
 - SPI_FLAG_FRE: Frame format error flag

Return value:

- The: new state of __FLAG__ (TRUE or FALSE).

[__HAL_SPI_CLEAR_CRCERRFLAG](#)**Description:**

- Clear the SPI CRCERR pending flag.

Parameters:

- __HANDLE__: specifies the SPI handle. This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.

Return value:

- None

[__HAL_SPI_CLEAR_MODFFLAG](#)**Description:**

- Clear the SPI MODF pending flag.

Parameters:

- __HANDLE__: specifies the SPI handle. This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.

Return value:

- None

[__HAL_SPI_CLEAR_OVRFAG](#)**Description:**

- Clear the SPI OVR pending flag.

Parameters:

- __HANDLE__: specifies the SPI handle. This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.

Return value:

- None

[__HAL_SPI_CLEAR_FREFLAG](#)**Description:**

- Clear the SPI FRE pending flag.

Parameters:

- `__HANDLE__`: specifies the SPI handle.
This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.

Return value:

- None

`__HAL_SPI_ENABLE`

Description:

- Enables the SPI.

Parameters:

- `__HANDLE__`: specifies the SPI Handle.
This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.

Return value:

- None

`__HAL_SPI_DISABLE`

Description:

- Disables the SPI.

Parameters:

- `__HANDLE__`: specifies the SPI Handle.
This parameter can be SPI where x: 1, 2, or 3 to select the SPI peripheral.

Return value:

- None

SPI Flag definition

`SPI_FLAG_RXNE`

`SPI_FLAG_TXE`

`SPI_FLAG_CRCERR`

`SPI_FLAG_MODF`

`SPI_FLAG_OVR`

`SPI_FLAG_BSY`

`SPI_FLAG_FRE`

SPI Interrupt configuration definition

`SPI_IT_TXE`

`SPI_IT_RXNE`

`SPI_IT_ERR`

SPI mode

`SPI_MODE_SLAVE`

`SPI_MODE_MASTER`

IS_SPI_MODE

SPI MSB LSB transmission

SPI_FIRSTBIT_MSB

SPI_FIRSTBIT_LSB

IS_SPI_FIRST_BIT

SPI Slave Select management

SPI_NSS_SOFT

SPI_NSS_HARD_INPUT

SPI_NSS_HARD_OUTPUT

IS_SPI_NSS

SPI TI mode

SPI_TIMODE_DISABLE

SPI_TIMODE_ENABLE

IS_SPI_TIMODE

41 HAL SPI Extension Driver

41.1 SPIEx Firmware driver defines

41.1.1 SPIEx

42 HAL SRAM Generic Driver

42.1 SRAM Firmware driver registers structures

42.1.1 SRAM_HandleTypeDef

Data Fields

- *FSMC_NORSRAM_TypeDef * Instance*
- *FSMC_NORSRAM_EXTENDED_TypeDef * Extended*
- *FSMC_NORSRAM_InitTypeDef Init*
- *HAL_LockTypeDef Lock*
- *__IO HAL_SRAM_StateTypeDef State*
- *DMA_HandleTypeDef * hdma*

Field Documentation

- *FSMC_NORSRAM_TypeDef* SRAM_HandleTypeDef::Instance*
Register base address
- *FSMC_NORSRAM_EXTENDED_TypeDef* SRAM_HandleTypeDef::Extended*
Extended mode register base address
- *FSMC_NORSRAM_InitTypeDef SRAM_HandleTypeDef::Init*
SRAM device control configuration parameters
- *HAL_LockTypeDef SRAM_HandleTypeDef::Lock*
SRAM locking object
- *__IO HAL_SRAM_StateTypeDef SRAM_HandleTypeDef::State*
SRAM device access state
- *DMA_HandleTypeDef* SRAM_HandleTypeDef::hdma*
Pointer DMA handler

42.2 SRAM Firmware driver API description

42.2.1 How to use this driver

This driver is a generic layered driver which contains a set of APIs used to control SRAM memories. It uses the FSMC layer functions to interface with SRAM devices. The following sequence should be followed to configure the FSMC to interface with SRAM/PSRAM memories:

1. Declare a SRAM_HandleTypeDef handle structure, for example:
SRAM_HandleTypeDef hsramp; and:
 - Fill the SRAM_HandleTypeDef handle "Init" field with the allowed values of the structure member.
 - Fill the SRAM_HandleTypeDef handle "Instance" field with a predefined base register instance for NOR or SRAM device
 - Fill the SRAM_HandleTypeDef handle "Extended" field with a predefined base register instance for NOR or SRAM extended mode
2. Declare two FSMC_NORSRAM_TimingTypeDef structures, for both normal and extended mode timings; for example: *FSMC_NORSRAM_TimingTypeDef Timing* and *FSMC_NORSRAM_TimingTypeDef ExTiming*; and fill its fields with the allowed values of the structure member.
3. Initialize the SRAM Controller by calling the function *HAL_SRAM_Init()*. This function performs the following sequence:
 - a. MSP hardware layer configuration using the function *HAL_SRAM_MspInit()*

- b. Control register configuration using the FSMC NORSRAM interface function `FSMC_NORSRAM_Init()`
 - c. Timing register configuration using the FSMC NORSRAM interface function `FSMC_NORSRAM_Timing_Init()`
 - d. Extended mode Timing register configuration using the FSMC NORSRAM interface function `FSMC_NORSRAM_Extended_Timing_Init()`
 - e. Enable the SRAM device using the macro `__FMC_NORSRAM_ENABLE()`
4. At this stage you can perform read/write accesses from/to the memory connected to the NOR/SRAM Bank. You can perform either polling or DMA transfer using the following APIs:
- `HAL_SRAM_Read()`/`HAL_SRAM_Write()` for polling read/write access
 - `HAL_SRAM_Read_DMA()`/`HAL_SRAM_Write_DMA()` for DMA read/write transfer
5. You can also control the SRAM device by calling the control APIs `HAL_SRAM_WriteOperation_Enable()`/`HAL_SRAM_WriteOperation_Disable()` to respectively enable/disable the SRAM write operation
6. You can continuously monitor the SRAM device HAL state by calling the function `HAL_SRAM_GetState()`

42.2.2 SRAM Initialization and de_initialization functions

This section provides functions allowing to initialize/de-initialize the SRAM memory

This section contains the following APIs:

- `HAL_SRAM_Init()`
- `HAL_SRAM_DeInit()`
- `HAL_SRAM_MspInit()`
- `HAL_SRAM_MspDeInit()`
- `HAL_SRAM_DMA_XferCpltCallback()`
- `HAL_SRAM_DMA_XferErrorCallback()`

42.2.3 SRAM Input and Output functions

This section provides functions allowing to use and control the SRAM memory

This section contains the following APIs:

- `HAL_SRAM_Read_8b()`
- `HAL_SRAM_Write_8b()`
- `HAL_SRAM_Read_16b()`
- `HAL_SRAM_Write_16b()`
- `HAL_SRAM_Read_32b()`
- `HAL_SRAM_Write_32b()`
- `HAL_SRAM_Read_DMA()`
- `HAL_SRAM_Write_DMA()`

42.2.4 SRAM Control functions

This subsection provides a set of functions allowing to control dynamically the SRAM interface.

This section contains the following APIs:

- `HAL_SRAM_WriteOperation_Enable()`
- `HAL_SRAM_WriteOperation_Disable()`

42.2.5 SRAM State functions

This subsection permits to get in run-time the status of the SRAM controller and the data flow.

This section contains the following APIs:

- [*HAL_SRAM_GetState\(\)*](#)

42.2.6 Detailed description of functions

HAL_SRAM_Init

Function name	HAL_StatusTypeDef HAL_SRAM_Init (SRAM_HandleTypeDef * hsram, FSMC_NORSRAM_TimingTypeDef * Timing, FSMC_NORSRAM_TimingTypeDef * ExtTiming)
Function description	Performs the SRAM device initialization sequence.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module. • Timing: Pointer to SRAM control timing structure • ExtTiming: Pointer to SRAM extended mode timing structure
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SRAM_DelInit

Function name	HAL_StatusTypeDef HAL_SRAM_DelInit (SRAM_HandleTypeDef * hsram)
Function description	Performs the SRAM device De-initialization sequence.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SRAM_MspInit

Function name	void HAL_SRAM_MspInit (SRAM_HandleTypeDef * hsram)
Function description	SRAM MSP Init.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SRAM_MspDelInit

Function name	void HAL_SRAM_MspDelInit (SRAM_HandleTypeDef * hsram)
Function description	SRAM MSP DelInit.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SRAM_DMA_XferCpltCallback

Function name	void HAL_SRAM_DMA_XferCpltCallback (DMA_HandleTypeDef * hdma)
Function description	DMA transfer complete callback.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SRAM_DMA_XferErrorCallback

Function name	void HAL_SRAM_DMA_XferErrorCallback (DMA_HandleTypeDef * hdma)
Function description	DMA transfer complete error callback.
Parameters	<ul style="list-style-type: none"> • hdma: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
Return values	<ul style="list-style-type: none"> • None:

HAL_SRAM_Read_8b

Function name	HAL_StatusTypeDef HAL_SRAM_Read_8b (SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint8_t * pDstBuffer, uint32_t BufferSize)
Function description	Reads 8-bit buffer from SRAM memory.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module. • pAddress: Pointer to read start address • pDstBuffer: Pointer to destination buffer • BufferSize: Size of the buffer to read from memory
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SRAM_Write_8b

Function name	HAL_StatusTypeDef HAL_SRAM_Write_8b (SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint8_t * pSrcBuffer, uint32_t BufferSize)
Function description	Writes 8-bit buffer to SRAM memory.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module. • pAddress: Pointer to write start address • pSrcBuffer: Pointer to source buffer to write • BufferSize: Size of the buffer to write to memory
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SRAM_Read_16b

Function name	HAL_StatusTypeDef HAL_SRAM_Read_16b (SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint16_t
---------------	---

*** pDstBuffer, uint32_t BufferSize)**

Function description Reads 16-bit buffer from SRAM memory.

- Parameters
- **hsram:** pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
 - **pAddress:** Pointer to read start address
 - **pDstBuffer:** Pointer to destination buffer
 - **BufferSize:** Size of the buffer to read from memory

Return values

- **HAL:** status

HAL_SRAM_Write_16b

Function name **HAL_StatusTypeDef HAL_SRAM_Write_16b
(SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint16_t
* pSrcBuffer, uint32_t BufferSize)**

Function description Writes 16-bit buffer to SRAM memory.

- Parameters
- **hsram:** pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
 - **pAddress:** Pointer to write start address
 - **pSrcBuffer:** Pointer to source buffer to write
 - **BufferSize:** Size of the buffer to write to memory

Return values

- **HAL:** status

HAL_SRAM_Read_32b

Function name **HAL_StatusTypeDef HAL_SRAM_Read_32b
(SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint32_t
* pDstBuffer, uint32_t BufferSize)**

Function description Reads 32-bit buffer from SRAM memory.

- Parameters
- **hsram:** pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
 - **pAddress:** Pointer to read start address
 - **pDstBuffer:** Pointer to destination buffer
 - **BufferSize:** Size of the buffer to read from memory

Return values

- **HAL:** status

HAL_SRAM_Write_32b

Function name **HAL_StatusTypeDef HAL_SRAM_Write_32b
(SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint32_t
* pSrcBuffer, uint32_t BufferSize)**

Function description Writes 32-bit buffer to SRAM memory.

- Parameters
- **hsram:** pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
 - **pAddress:** Pointer to write start address
 - **pSrcBuffer:** Pointer to source buffer to write
 - **BufferSize:** Size of the buffer to write to memory

Return values

- **HAL:** status

HAL_SRAM_Read_DMA

Function name	HAL_StatusTypeDef HAL_SRAM_Read_DMA (SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint32_t * pDstBuffer, uint32_t BufferSize)
Function description	Reads a Words data from the SRAM memory using DMA transfer.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module. • pAddress: Pointer to read start address • pDstBuffer: Pointer to destination buffer • BufferSize: Size of the buffer to read from memory
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SRAM_Write_DMA

Function name	HAL_StatusTypeDef HAL_SRAM_Write_DMA (SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint32_t * pSrcBuffer, uint32_t BufferSize)
Function description	Writes a Words data buffer to SRAM memory using DMA transfer.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module. • pAddress: Pointer to write start address • pSrcBuffer: Pointer to source buffer to write • BufferSize: Size of the buffer to write to memory
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SRAM_WriteOperation_Enable

Function name	HAL_StatusTypeDef HAL_SRAM_WriteOperation_Enable (SRAM_HandleTypeDef * hsram)
Function description	Enables dynamically SRAM write operation.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SRAM_WriteOperation_Disable

Function name	HAL_StatusTypeDef HAL_SRAM_WriteOperation_Disable (SRAM_HandleTypeDef * hsram)
Function description	Disables dynamically SRAM write operation.
Parameters	<ul style="list-style-type: none"> • hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_SRAM_GetState

Function name	HAL_SRAM_StateTypeDef HAL_SRAM_GetState (SRAM_HandleTypeDef * hsram)
---------------	---

Function description	Returns the SRAM controller state.
Parameters	<ul style="list-style-type: none">• hsram: pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.
Return values	<ul style="list-style-type: none">• HAL: state

42.3 SRAM Firmware driver defines

42.3.1 SRAM

SRAM Exported Macros

`_HAL_SRAM_RESET_HANDLE_STATE` **Description:**

- Reset SRAM handle state.

Parameters:

- `_HANDLE_`: SRAM handle

Return value:

- None

43 HAL TIM Generic Driver

43.1 TIM Firmware driver registers structures

43.1.1 TIM_Base_InitTypeDef

Data Fields

- *uint32_t Prescaler*
- *uint32_t CounterMode*
- *uint32_t Period*
- *uint32_t ClockDivision*

Field Documentation

- ***uint32_t TIM_Base_InitTypeDef::Prescaler***
Specifies the prescaler value used to divide the TIM clock. This parameter can be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF
- ***uint32_t TIM_Base_InitTypeDef::CounterMode***
Specifies the counter mode. This parameter can be a value of [**TIM_Counter_Mode**](#)
- ***uint32_t TIM_Base_InitTypeDef::Period***
Specifies the period value to be loaded into the active Auto-Reload Register at the next update event. This parameter can be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF.
- ***uint32_t TIM_Base_InitTypeDef::ClockDivision***
Specifies the clock division. This parameter can be a value of [**TIM_ClockDivision**](#)

43.1.2 TIM_OC_InitTypeDef

Data Fields

- *uint32_t OCMode*
- *uint32_t Pulse*
- *uint32_t OCPolarity*
- *uint32_t OCFastMode*
- *uint32_t OCIdleState*

Field Documentation

- ***uint32_t TIM_OC_InitTypeDef::OCMode***
Specifies the TIM mode. This parameter can be a value of [**TIM_Output_Compare_and_PWM_modes**](#)
- ***uint32_t TIM_OC_InitTypeDef::Pulse***
Specifies the pulse value to be loaded into the Capture Compare Register. This parameter can be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF
- ***uint32_t TIM_OC_InitTypeDef::OCPolarity***
Specifies the output polarity. This parameter can be a value of [**TIM_Output_Compare_Polarity**](#)
- ***uint32_t TIM_OC_InitTypeDef::OCFastMode***
Specifies the Fast mode state. This parameter can be a value of [**TIM_Output_Fast_State**](#)
Note:This parameter is valid only in PWM1 and PWM2 mode.
- ***uint32_t TIM_OC_InitTypeDef::OCIdleState***
Specifies the TIM Output Compare pin state during Idle state. This parameter can be a value of [**TIM_Output_Compare_Idle_State**](#).

43.1.3 TIM_OnePulse_InitTypeDef

Data Fields

- *uint32_t OCMode*
- *uint32_t Pulse*
- *uint32_t OCPolarity*
- *uint32_t OCIdleState*
- *uint32_t ICPolarity*
- *uint32_t ICSelection*
- *uint32_t ICFilter*

Field Documentation

- ***uint32_t TIM_OnePulse_InitTypeDef::OCMode***
Specifies the TIM mode. This parameter can be a value of [*TIM_Output_Compare_and_PWM_modes*](#)
- ***uint32_t TIM_OnePulse_InitTypeDef::Pulse***
Specifies the pulse value to be loaded into the Capture Compare Register. This parameter can be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF
- ***uint32_t TIM_OnePulse_InitTypeDef::OCPolarity***
Specifies the output polarity. This parameter can be a value of [*TIM_Output_Compare_Polarity*](#)
- ***uint32_t TIM_OnePulse_InitTypeDef::OCIdleState***
Specifies the TIM Output Compare pin state during Idle state. This parameter can be a value of [*TIM_Output_Compare_Idle_State*](#).
- ***uint32_t TIM_OnePulse_InitTypeDef::ICPolarity***
Specifies the active edge of the input signal. This parameter can be a value of [*TIM_Input_Capture_Polarity*](#)
- ***uint32_t TIM_OnePulse_InitTypeDef::ICSelection***
Specifies the input. This parameter can be a value of [*TIM_Input_Capture_Selection*](#)
- ***uint32_t TIM_OnePulse_InitTypeDef::ICFilter***
Specifies the input capture filter. This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF

43.1.4 TIM_IC_InitTypeDef

Data Fields

- *uint32_t ICPolarity*
- *uint32_t ICSelection*
- *uint32_t ICPrescaler*
- *uint32_t ICFilter*

Field Documentation

- ***uint32_t TIM_IC_InitTypeDef::ICPolarity***
Specifies the active edge of the input signal. This parameter can be a value of [*TIM_Input_Capture_Polarity*](#)
- ***uint32_t TIM_IC_InitTypeDef::ICSelection***
Specifies the input. This parameter can be a value of [*TIM_Input_Capture_Selection*](#)
- ***uint32_t TIM_IC_InitTypeDef::ICPrescaler***
Specifies the Input Capture Prescaler. This parameter can be a value of [*TIM_Input_Capture_Prescaler*](#)
- ***uint32_t TIM_IC_InitTypeDef::ICFilter***
Specifies the input capture filter. This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF

43.1.5 TIM_Encoder_InitTypeDef

Data Fields

- `uint32_t EncoderMode`
- `uint32_t IC1Polarity`
- `uint32_t IC1Selection`
- `uint32_t IC1Prescaler`
- `uint32_t IC1Filter`
- `uint32_t IC2Polarity`
- `uint32_t IC2Selection`
- `uint32_t IC2Prescaler`
- `uint32_t IC2Filter`

Field Documentation

- **`uint32_t TIM_Encoder_InitTypeDef::EncoderMode`**
Specifies the active edge of the input signal. This parameter can be a value of [`TIM_Encoder_Mode`](#)
- **`uint32_t TIM_Encoder_InitTypeDef::IC1Polarity`**
Specifies the active edge of the input signal. This parameter can be a value of [`TIM_Input_Capture_Polarity`](#)
- **`uint32_t TIM_Encoder_InitTypeDef::IC1Selection`**
Specifies the input. This parameter can be a value of [`TIM_Input_Capture_Selection`](#)
- **`uint32_t TIM_Encoder_InitTypeDef::IC1Prescaler`**
Specifies the Input Capture Prescaler. This parameter can be a value of [`TIM_Input_Capture_Prescaler`](#)
- **`uint32_t TIM_Encoder_InitTypeDef::IC1Filter`**
Specifies the input capture filter. This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF
- **`uint32_t TIM_Encoder_InitTypeDef::IC2Polarity`**
Specifies the active edge of the input signal. This parameter can be a value of [`TIM_Input_Capture_Polarity`](#)
- **`uint32_t TIM_Encoder_InitTypeDef::IC2Selection`**
Specifies the input. This parameter can be a value of [`TIM_Input_Capture_Selection`](#)
- **`uint32_t TIM_Encoder_InitTypeDef::IC2Prescaler`**
Specifies the Input Capture Prescaler. This parameter can be a value of [`TIM_Input_Capture_Prescaler`](#)
- **`uint32_t TIM_Encoder_InitTypeDef::IC2Filter`**
Specifies the input capture filter. This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF

43.1.6 TIM_ClockConfigTypeDef

Data Fields

- `uint32_t ClockSource`
- `uint32_t ClockPolarity`
- `uint32_t ClockPrescaler`
- `uint32_t ClockFilter`

Field Documentation

- **`uint32_t TIM_ClockConfigTypeDef::ClockSource`**
TIM clock sources This parameter can be a value of [`TIM_Clock_Source`](#)
- **`uint32_t TIM_ClockConfigTypeDef::ClockPolarity`**
TIM clock polarity This parameter can be a value of [`TIM_Clock_Polarity`](#)

- ***uint32_t TIM_ClockConfigTypeDef::ClockPrescaler***
TIM clock prescaler This parameter can be a value of [*TIM_Clock_Prescaler*](#)
- ***uint32_t TIM_ClockConfigTypeDef::ClockFilter***
TIM clock filter This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF

43.1.7 TIM_ClearInputConfigTypeDef

Data Fields

- ***uint32_t ClearInputState***
- ***uint32_t ClearInputSource***
- ***uint32_t ClearInputPolarity***
- ***uint32_t ClearInputPrescaler***
- ***uint32_t ClearInputFilter***

Field Documentation

- ***uint32_t TIM_ClearInputConfigTypeDef::ClearInputState***
TIM clear Input state This parameter can be ENABLE or DISABLE
- ***uint32_t TIM_ClearInputConfigTypeDef::ClearInputSource***
TIM clear Input sources This parameter can be a value of [*TIM_ClearInput_Source*](#)
- ***uint32_t TIM_ClearInputConfigTypeDef::ClearInputPolarity***
TIM Clear Input polarity This parameter can be a value of [*TIM_ClearInput_Polarity*](#)
- ***uint32_t TIM_ClearInputConfigTypeDef::ClearInputPrescaler***
TIM Clear Input prescaler This parameter can be a value of [*TIM_ClearInput_Prescaler*](#)
- ***uint32_t TIM_ClearInputConfigTypeDef::ClearInputFilter***
TIM Clear Input filter This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF

43.1.8 TIM_SlaveConfigTypeDef

Data Fields

- ***uint32_t SlaveMode***
- ***uint32_t InputTrigger***
- ***uint32_t TriggerPolarity***
- ***uint32_t TriggerPrescaler***
- ***uint32_t TriggerFilter***

Field Documentation

- ***uint32_t TIM_SlaveConfigTypeDef::SlaveMode***
Slave mode selection This parameter can be a value of [*TIM_Slave_Mode*](#)
- ***uint32_t TIM_SlaveConfigTypeDef::InputTrigger***
Input Trigger source This parameter can be a value of [*TIM_Trigger_Selection*](#)
- ***uint32_t TIM_SlaveConfigTypeDef::TriggerPolarity***
Input Trigger polarity This parameter can be a value of [*TIM_Trigger_Polarity*](#)
- ***uint32_t TIM_SlaveConfigTypeDef::TriggerPrescaler***
Input trigger prescaler This parameter can be a value of [*TIM_Trigger_Prescaler*](#)
- ***uint32_t TIM_SlaveConfigTypeDef::TriggerFilter***
Input trigger filter This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF

43.1.9 TIM_HandleTypeDef

Data Fields

- *TIM_TypeDef * Instance*
- *TIM_Base_InitTypeDef Init*
- *HAL_TIM_ActiveChannel Channel*
- *DMA_HandleTypeDef * hdma*
- *HAL_LockTypeDef Lock*
- *__IO HAL_TIM_StateTypeDef State*

Field Documentation

- *TIM_TypeDef* TIM_HandleTypeDef::Instance*
Register base address
- *TIM_Base_InitTypeDef TIM_HandleTypeDef::Init*
TIM Time Base required parameters
- *HAL_TIM_ActiveChannel TIM_HandleTypeDef::Channel*
Active channel
- *DMA_HandleTypeDef* TIM_HandleTypeDef::hdma[7]*
DMA Handlers array This array is accessed by a *TIM_DMA_Handle_index*
- *HAL_LockTypeDef TIM_HandleTypeDef::Lock*
Locking object
- *__IO HAL_TIM_StateTypeDef TIM_HandleTypeDef::State*
TIM operation state

43.2 TIM Firmware driver API description

43.2.1 TIMER Generic features

The Timer features include:

1. 16-bit up, down, up/down auto-reload counter.
2. 16-bit programmable prescaler allowing dividing (also on the fly) the counter clock frequency either by any factor between 1 and 65536.
3. Up to 4 independent channels for:
 - Input Capture
 - Output Compare
 - PWM generation (Edge and Center-aligned Mode)
 - One-pulse mode output
4. Synchronization circuit to control the timer with external signals and to interconnect several timers together.
5. Supports incremental (quadrature) encoder

43.2.2 How to use this driver

1. Initialize the TIM low level resources by implementing the following functions depending from feature used :
 - Time Base : `HAL_TIM_Base_MspInit()`
 - Input Capture : `HAL_TIM_IC_MspInit()`
 - Output Compare : `HAL_TIM_OC_MspInit()`
 - PWM generation : `HAL_TIM_PWM_MspInit()`
 - One-pulse mode output : `HAL_TIM_OnePulse_MspInit()`
 - Encoder mode output : `HAL_TIM_Encoder_MspInit()`
2. Initialize the TIM low level resources :
 - a. Enable the TIM interface clock using `__HAL_RCC_TIMx_CLK_ENABLE()`;
 - b. TIM pins configuration
 - Enable the clock for the TIM GPIOs using the following function:
`__HAL_RCC_GPIOx_CLK_ENABLE();`
 - Configure these TIM pins in Alternate function mode using `HAL_GPIO_Init()`;

3. The external Clock can be configured, if needed (the default clock is the internal clock from the APBx), using the following function: HAL_TIM_ConfigClockSource, the clock configuration should be done before any start function.
4. Configure the TIM in the desired functioning mode using one of the Initialization function of this driver:
 - HAL_TIM_Base_Init: to use the Timer to generate a simple time base
 - HAL_TIM_OC_Init and HAL_TIM_OC_ConfigChannel: to use the Timer to generate an Output Compare signal.
 - HAL_TIM_PWM_Init and HAL_TIM_PWM_ConfigChannel: to use the Timer to generate a PWM signal.
 - HAL_TIM_IC_Init and HAL_TIM_IC_ConfigChannel: to use the Timer to measure an external signal.
 - HAL_TIM_OnePulse_Init and HAL_TIM_OnePulse_ConfigChannel: to use the Timer in One Pulse Mode.
 - HAL_TIM_Encoder_Init: to use the Timer Encoder Interface.
5. Activate the TIM peripheral using one of the start functions depending from the feature used:
 - Time Base : HAL_TIM_Base_Start(), HAL_TIM_Base_Start_DMA(), HAL_TIM_Base_Start_IT()
 - Input Capture : HAL_TIM_IC_Start(), HAL_TIM_IC_Start_DMA(), HAL_TIM_IC_Start_IT()
 - Output Compare : HAL_TIM_OC_Start(), HAL_TIM_OC_Start_DMA(), HAL_TIM_OC_Start_IT()
 - PWM generation : HAL_TIM_PWM_Start(), HAL_TIM_PWM_Start_DMA(), HAL_TIM_PWM_Start_IT()
 - One-pulse mode output : HAL_TIM_OnePulse_Start(), HAL_TIM_OnePulse_Start_IT()
 - Encoder mode output : HAL_TIM_Encoder_Start(), HAL_TIM_Encoder_Start_DMA(), HAL_TIM_Encoder_Start_IT().
6. The DMA Burst is managed with the two following functions:
HAL_TIM_DMABurst_WriteStart() HAL_TIM_DMABurst_ReadStart()

43.2.3 Time Base functions

This section provides functions allowing to:

- Initialize and configure the TIM base.
- De-initialize the TIM base.
- Start the Time Base.
- Stop the Time Base.
- Start the Time Base and enable interrupt.
- Stop the Time Base and disable interrupt.
- Start the Time Base and enable DMA transfer.
- Stop the Time Base and disable DMA transfer.

This section contains the following APIs:

- [**HAL_TIM_Base_Init\(\)**](#)
- [**HAL_TIM_Base_DelInit\(\)**](#)
- [**HAL_TIM_Base_MspInit\(\)**](#)
- [**HAL_TIM_Base_MspDelInit\(\)**](#)
- [**HAL_TIM_Base_Start\(\)**](#)
- [**HAL_TIM_Base_Stop\(\)**](#)
- [**HAL_TIM_Base_Start_IT\(\)**](#)
- [**HAL_TIM_Base_Stop_IT\(\)**](#)
- [**HAL_TIM_Base_Start_DMA\(\)**](#)

- [*HAL_TIM_Base_Stop_DMA\(\)*](#)

43.2.4 Time Output Compare functions

This section provides functions allowing to:

- Initialize and configure the TIM Output Compare.
- De-initialize the TIM Output Compare.
- Start the Time Output Compare.
- Stop the Time Output Compare.
- Start the Time Output Compare and enable interrupt.
- Stop the Time Output Compare and disable interrupt.
- Start the Time Output Compare and enable DMA transfer.
- Stop the Time Output Compare and disable DMA transfer.

This section contains the following APIs:

- [*HAL_TIM_OC_Init\(\)*](#)
- [*HAL_TIM_OC_DeInit\(\)*](#)
- [*HAL_TIM_OC_MspInit\(\)*](#)
- [*HAL_TIM_OC_MspDeInit\(\)*](#)
- [*HAL_TIM_OC_Start\(\)*](#)
- [*HAL_TIM_OC_Stop\(\)*](#)
- [*HAL_TIM_OC_Start_IT\(\)*](#)
- [*HAL_TIM_OC_Stop_IT\(\)*](#)
- [*HAL_TIM_OC_Start_DMA\(\)*](#)
- [*HAL_TIM_OC_Stop_DMA\(\)*](#)

43.2.5 Time PWM functions

This section provides functions allowing to:

- Initialize and configure the TIM PWM.
- De-initialize the TIM PWM.
- Start the Time PWM.
- Stop the Time PWM.
- Start the Time PWM and enable interrupt.
- Stop the Time PWM and disable interrupt.
- Start the Time PWM and enable DMA transfer.
- Stop the Time PWM and disable DMA transfer.

This section contains the following APIs:

- [*HAL_TIM_PWM_Init\(\)*](#)
- [*HAL_TIM_PWM_DeInit\(\)*](#)
- [*HAL_TIM_PWM_MspInit\(\)*](#)
- [*HAL_TIM_PWM_MspDeInit\(\)*](#)
- [*HAL_TIM_PWM_Start\(\)*](#)
- [*HAL_TIM_PWM_Stop\(\)*](#)
- [*HAL_TIM_PWM_Start_IT\(\)*](#)
- [*HAL_TIM_PWM_Stop_IT\(\)*](#)
- [*HAL_TIM_PWM_Start_DMA\(\)*](#)
- [*HAL_TIM_PWM_Stop_DMA\(\)*](#)

43.2.6 Time Input Capture functions

This section provides functions allowing to:

- Initialize and configure the TIM Input Capture.
- De-initialize the TIM Input Capture.
- Start the Time Input Capture.
- Stop the Time Input Capture.
- Start the Time Input Capture and enable interrupt.
- Stop the Time Input Capture and disable interrupt.
- Start the Time Input Capture and enable DMA transfer.
- Stop the Time Input Capture and disable DMA transfer.

This section contains the following APIs:

- [*HAL_TIM_IC_Init\(\)*](#)
- [*HAL_TIM_IC_DeInit\(\)*](#)
- [*HAL_TIM_IC_MspInit\(\)*](#)
- [*HAL_TIM_IC_MspDeInit\(\)*](#)
- [*HAL_TIM_IC_Start\(\)*](#)
- [*HAL_TIM_IC_Stop\(\)*](#)
- [*HAL_TIM_IC_Start_IT\(\)*](#)
- [*HAL_TIM_IC_Stop_IT\(\)*](#)
- [*HAL_TIM_IC_Start_DMA\(\)*](#)
- [*HAL_TIM_IC_Stop_DMA\(\)*](#)

43.2.7 Time One Pulse functions

This section provides functions allowing to:

- Initialize and configure the TIM One Pulse.
- De-initialize the TIM One Pulse.
- Start the Time One Pulse.
- Stop the Time One Pulse.
- Start the Time One Pulse and enable interrupt.
- Stop the Time One Pulse and disable interrupt.
- Start the Time One Pulse and enable DMA transfer.
- Stop the Time One Pulse and disable DMA transfer.

This section contains the following APIs:

- [*HAL_TIM_OnePulse_Init\(\)*](#)
- [*HAL_TIM_OnePulse_DeInit\(\)*](#)
- [*HAL_TIM_OnePulse_MspInit\(\)*](#)
- [*HAL_TIM_OnePulse_MspDeInit\(\)*](#)
- [*HAL_TIM_OnePulse_Start\(\)*](#)
- [*HAL_TIM_OnePulse_Stop\(\)*](#)
- [*HAL_TIM_OnePulse_Start_IT\(\)*](#)
- [*HAL_TIM_OnePulse_Stop_IT\(\)*](#)

43.2.8 Time Encoder functions

This section provides functions allowing to:

- Initialize and configure the TIM Encoder.
- De-initialize the TIM Encoder.
- Start the Time Encoder.
- Stop the Time Encoder.
- Start the Time Encoder and enable interrupt.
- Stop the Time Encoder and disable interrupt.
- Start the Time Encoder and enable DMA transfer.

- Stop the Time Encoder and disable DMA transfer.

This section contains the following APIs:

- [*HAL_TIM_Encoder_Init\(\)*](#)
- [*HAL_TIM_Encoder_DelInit\(\)*](#)
- [*HAL_TIM_Encoder_MspInit\(\)*](#)
- [*HAL_TIM_Encoder_MspDelInit\(\)*](#)
- [*HAL_TIM_Encoder_Start\(\)*](#)
- [*HAL_TIM_Encoder_Stop\(\)*](#)
- [*HAL_TIM_Encoder_Start_IT\(\)*](#)
- [*HAL_TIM_Encoder_Stop_IT\(\)*](#)
- [*HAL_TIM_Encoder_Start_DMA\(\)*](#)
- [*HAL_TIM_Encoder_Stop_DMA\(\)*](#)

43.2.9 IRQ handler management

This section provides Timer IRQ handler function.

This section contains the following APIs:

- [*HAL_TIM_IRQHandler\(\)*](#)

43.2.10 Peripheral Control functions

This section provides functions allowing to:

- Configure The Input Output channels for OC, PWM, IC or One Pulse mode.
- Configure External Clock source.
- Configure Complementary channels, break features and dead time.
- Configure Master and the Slave synchronization.
- Configure the DMA Burst Mode.

This section contains the following APIs:

- [*HAL_TIM_OC_ConfigChannel\(\)*](#)
- [*HAL_TIM_IC_ConfigChannel\(\)*](#)
- [*HAL_TIM_PWM_ConfigChannel\(\)*](#)
- [*HAL_TIM_OnePulse_ConfigChannel\(\)*](#)
- [*HAL_TIM_DMABurst_WriteStart\(\)*](#)
- [*HAL_TIM_DMABurst_WriteStop\(\)*](#)
- [*HAL_TIM_DMABurst_ReadStart\(\)*](#)
- [*HAL_TIM_DMABurst_ReadStop\(\)*](#)
- [*HAL_TIM_GenerateEvent\(\)*](#)
- [*HAL_TIM_ConfigOCrefClear\(\)*](#)
- [*HAL_TIM_ConfigClockSource\(\)*](#)
- [*HAL_TIM_ConfigTI1Input\(\)*](#)
- [*HAL_TIM_SlaveConfigSynchronization\(\)*](#)
- [*HAL_TIM_SlaveConfigSynchronization_IT\(\)*](#)
- [*HAL_TIM_ReadCapturedValue\(\)*](#)

43.2.11 TIM Callbacks functions

This section provides TIM callback functions:

- Timer Period elapsed callback
- Timer Output Compare callback
- Timer Input capture callback

- Timer Trigger callback
- Timer Error callback

This section contains the following APIs:

- [*HAL_TIM_PeriodElapsedCallback\(\)*](#)
- [*HAL_TIM_OC_DelayElapsedCallback\(\)*](#)
- [*HAL_TIM_IC_CaptureCallback\(\)*](#)
- [*HAL_TIM_PWM_PulseFinishedCallback\(\)*](#)
- [*HAL_TIM_TriggerCallback\(\)*](#)
- [*HAL_TIM_ErrorCallback\(\)*](#)

43.2.12 Peripheral State functions

This subsection permit to get in run-time the status of the peripheral and the data flow.

This section contains the following APIs:

- [*HAL_TIM_Base_GetState\(\)*](#)
- [*HAL_TIM_OC_GetState\(\)*](#)
- [*HAL_TIM_PWM_GetState\(\)*](#)
- [*HAL_TIM_IC_GetState\(\)*](#)
- [*HAL_TIM_OnePulse_GetState\(\)*](#)
- [*HAL_TIM_Encoder_GetState\(\)*](#)
- [*TIM_DMAError\(\)*](#)
- [*TIM_DMADelayPulseCplt\(\)*](#)
- [*TIM_DMACaptureCplt\(\)*](#)

43.2.13 Detailed description of functions

HAL_TIM_Base_Init

Function name	HAL_StatusTypeDef HAL_TIM_Base_Init (TIM_HandleTypeDef * htim)
Function description	Initializes the TIM Time base Unit according to the specified parameters in the TIM_HandleTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • htim: TIM Base handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Base_DeInit

Function name	HAL_StatusTypeDef HAL_TIM_Base_DeInit (TIM_HandleTypeDef * htim)
Function description	DeInitializes the TIM Base peripheral.
Parameters	<ul style="list-style-type: none"> • htim: TIM Base handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Base_MspInit

Function name	void HAL_TIM_Base_MspInit (TIM_HandleTypeDef * htim)
Function description	Initializes the TIM Base MSP.

Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_Base_MspDeInit

Function name	void HAL_TIM_Base_MspDeInit (TIM_HandleTypeDef * htim)
Function description	Deinitializes TIM Base MSP.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_Base_Start

Function name	HAL_StatusTypeDef HAL_TIM_Base_Start (TIM_HandleTypeDef * htim)
Function description	Starts the TIM Base generation.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Base_Stop

Function name	HAL_StatusTypeDef HAL_TIM_Base_Stop (TIM_HandleTypeDef * htim)
Function description	Stops the TIM Base generation.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Base_Start_IT

Function name	HAL_StatusTypeDef HAL_TIM_Base_Start_IT (TIM_HandleTypeDef * htim)
Function description	Starts the TIM Base generation in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Base_Stop_IT

Function name	HAL_StatusTypeDef HAL_TIM_Base_Stop_IT (TIM_HandleTypeDef * htim)
Function description	Stops the TIM Base generation in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Base_Start_DMA

Function name	HAL_StatusTypeDef HAL_TIM_Base_Start_DMA
---------------	---

(TIM_HandleTypeDef * htim, uint32_t * pData, uint16_t Length)

Function description Starts the TIM Base generation in DMA mode.

Parameters

- **htim:** : TIM handle
- **pData:** The source Buffer address.
- **Length:** The length of data to be transferred from memory to peripheral.

Return values

- **HAL:** status

HAL_TIM_Base_Stop_DMA

Function name **HAL_StatusTypeDef HAL_TIM_Base_Stop_DMA (TIM_HandleTypeDef * htim)**

Function description Stops the TIM Base generation in DMA mode.

Parameters

- **htim:** : TIM handle

Return values

- **HAL:** status

HAL_TIM_OC_Init

Function name **HAL_StatusTypeDef HAL_TIM_OC_Init (TIM_HandleTypeDef * htim)**

Function description Initializes the TIM Output Compare according to the specified parameters in the TIM_HandleTypeDef and create the associated handle.

Parameters

- **htim:** TIM Output Compare handle

Return values

- **HAL:** status

HAL_TIM_OC_DelInit

Function name **HAL_StatusTypeDef HAL_TIM_OC_DelInit (TIM_HandleTypeDef * htim)**

Function description Delinitializes the TIM peripheral.

Parameters

- **htim:** TIM Output Compare handle

Return values

- **HAL:** status

HAL_TIM_OC_MspInit

Function name **void HAL_TIM_OC_MspInit (TIM_HandleTypeDef * htim)**

Function description Initializes the TIM Output Compare MSP.

Parameters

- **htim:** TIM handle

Return values

- **None:**

HAL_TIM_OC_MspDelInit

Function name **void HAL_TIM_OC_MspDelInit (TIM_HandleTypeDef * htim)**

Function description Delinitializes TIM Output Compare MSP.

Parameters	<ul style="list-style-type: none"> htim: TIM handle
Return values	<ul style="list-style-type: none"> None:

HAL_TIM_OC_Start

Function name	HAL_StatusTypeDef HAL_TIM_OC_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Starts the TIM Output Compare signal generation.
Parameters	<ul style="list-style-type: none"> htim: : TIM Output Compare handle Channel: : TIM Channel to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_OC_Stop

Function name	HAL_StatusTypeDef HAL_TIM_OC_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the TIM Output Compare signal generation.
Parameters	<ul style="list-style-type: none"> htim: : TIM handle Channel: : TIM Channel to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_OC_Start_IT

Function name	HAL_StatusTypeDef HAL_TIM_OC_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Starts the TIM Output Compare signal generation in interrupt mode.
Parameters	<ul style="list-style-type: none"> htim: : TIM OC handle Channel: : TIM Channel to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_OC_Stop_IT

Function name	HAL_StatusTypeDef HAL_TIM_OC_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the TIM Output Compare signal generation in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Output Compare handle • Channel: : TIM Channel to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_OC_Start_DMA

Function name	HAL_StatusTypeDef HAL_TIM_OC_Start_DMA (TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData, uint16_t Length)
Function description	Starts the TIM Output Compare signal generation in DMA mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Output Compare handle • Channel: : TIM Channel to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected • pData: The source Buffer address. • Length: The length of data to be transferred from memory to TIM peripheral
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_OC_Stop_DMA

Function name	HAL_StatusTypeDef HAL_TIM_OC_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the TIM Output Compare signal generation in DMA mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Output Compare handle • Channel: : TIM Channel to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_PWM_Init

Function name	HAL_StatusTypeDef HAL_TIM_PWM_Init (TIM_HandleTypeDef * htim)
Function description	Initializes the TIM PWM Time Base according to the specified parameters in the TIM_HandleTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_PWM_DeInit

Function name	HAL_StatusTypeDef HAL_TIM_PWM_DeInit (TIM_HandleTypeDef * htim)
Function description	DeInitializes the TIM peripheral.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_PWM_MspInit

Function name	void HAL_TIM_PWM_MspInit (TIM_HandleTypeDef * htim)
Function description	Initializes the TIM PWM MSP.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_PWM_MspDeInit

Function name	void HAL_TIM_PWM_MspDeInit (TIM_HandleTypeDef * htim)
Function description	DeInitializes TIM PWM MSP.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_PWM_Start

Function name	HAL_StatusTypeDef HAL_TIM_PWM_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Starts the PWM signal generation.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle • Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_PWM_Stop

Function name	HAL_StatusTypeDef HAL_TIM_PWM_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the PWM signal generation.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle • Channel: : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_PWM_Start_IT

Function name	HAL_StatusTypeDef HAL_TIM_PWM_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Starts the PWM signal generation in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle • Channel: : TIM Channel to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_PWM_Stop_IT

Function name	HAL_StatusTypeDef HAL_TIM_PWM_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the PWM signal generation in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle • Channel: : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_PWM_Start_DMA

Function name	HAL_StatusTypeDef HAL_TIM_PWM_Start_DMA (TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData, uint16_t Length)
Function description	Starts the TIM PWM signal generation in DMA mode.

Parameters	<ul style="list-style-type: none"> htim: : TIM handle Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected pData: The source Buffer address. Length: The length of data to be transferred from memory to TIM peripheral
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_PWM_Stop_DMA

Function name	HAL_StatusTypeDef HAL_TIM_PWM_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the TIM PWM signal generation in DMA mode.
Parameters	<ul style="list-style-type: none"> htim: : TIM handle Channel: : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_IC_Init

Function name	HAL_StatusTypeDef HAL_TIM_IC_Init (TIM_HandleTypeDef * htim)
Function description	Initializes the TIM Input Capture Time base according to the specified parameters in the TIM_HandleTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> htim: TIM Input Capture handle
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_IC_DeInit

Function name	HAL_StatusTypeDef HAL_TIM_IC_DeInit (TIM_HandleTypeDef * htim)
Function description	Deinitializes the TIM peripheral.
Parameters	<ul style="list-style-type: none"> htim: TIM Input Capture handle
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_IC_MspInit

Function name	void HAL_TIM_IC_MspInit (TIM_HandleTypeDef * htim)
---------------	---

Function description	Initializes the TIM Input Capture MSP.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_IC_MspDeInit

Function name	void HAL_TIM_IC_MspDeInit (TIM_HandleTypeDef * htim)
Function description	Deinitializes TIM Input Capture MSP.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_IC_Start

Function name	HAL_StatusTypeDef HAL_TIM_IC_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Starts the TIM Input Capture measurement.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Input Capture handle • Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_IC_Stop

Function name	HAL_StatusTypeDef HAL_TIM_IC_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the TIM Input Capture measurement.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle • Channel: : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_IC_Start_IT

Function name	HAL_StatusTypeDef HAL_TIM_IC_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Starts the TIM Input Capture measurement in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Input Capture handle • Channel: : TIM Channels to be enabled This parameter can be one of the following values:

- TIM_CHANNEL_1: TIM Channel 1 selected
- TIM_CHANNEL_2: TIM Channel 2 selected
- TIM_CHANNEL_3: TIM Channel 3 selected
- TIM_CHANNEL_4: TIM Channel 4 selected

Return values

- **HAL:** status

HAL_TIM_IC_Stop_IT

Function name

**HAL_StatusTypeDef HAL_TIM_IC_Stop_IT
(TIM_HandleTypeDef * htim, uint32_t Channel)**

Function description

Stops the TIM Input Capture measurement in interrupt mode.

Parameters

- **htim:** : TIM handle
- **Channel:** : TIM Channels to be disabled This parameter can be one of the following values:
 - TIM_CHANNEL_1: TIM Channel 1 selected
 - TIM_CHANNEL_2: TIM Channel 2 selected
 - TIM_CHANNEL_3: TIM Channel 3 selected
 - TIM_CHANNEL_4: TIM Channel 4 selected

Return values

- **HAL:** status

HAL_TIM_IC_Start_DMA

Function name

**HAL_StatusTypeDef HAL_TIM_IC_Start_DMA
(TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t *
pData, uint16_t Length)**

Function description

Starts the TIM Input Capture measurement in DMA mode.

Parameters

- **htim:** : TIM Input Capture handle
- **Channel:** : TIM Channels to be enabled This parameter can be one of the following values:
 - TIM_CHANNEL_1: TIM Channel 1 selected
 - TIM_CHANNEL_2: TIM Channel 2 selected
 - TIM_CHANNEL_3: TIM Channel 3 selected
 - TIM_CHANNEL_4: TIM Channel 4 selected
- **pData:** The destination Buffer address.
- **Length:** The length of data to be transferred from TIM peripheral to memory.

Return values

- **HAL:** status

HAL_TIM_IC_Stop_DMA

Function name

**HAL_StatusTypeDef HAL_TIM_IC_Stop_DMA
(TIM_HandleTypeDef * htim, uint32_t Channel)**

Function description

Stops the TIM Input Capture measurement in DMA mode.

Parameters

- **htim:** : TIM Input Capture handle
- **Channel:** : TIM Channels to be disabled This parameter can be one of the following values:
 - TIM_CHANNEL_1: TIM Channel 1 selected
 - TIM_CHANNEL_2: TIM Channel 2 selected
 - TIM_CHANNEL_3: TIM Channel 3 selected

– TIM_CHANNEL_4: TIM Channel 4 selected

Return values • HAL: status

HAL_TIM_OnePulse_Init

Function name	HAL_StatusTypeDef HAL_TIM_OnePulse_Init (TIM_HandleTypeDef * htim, uint32_t OnePulseMode)
Function description	Initializes the TIM One Pulse Time Base according to the specified parameters in the TIM_HandleTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • htim: TIM OnePulse handle • OnePulseMode: Select the One pulse mode. This parameter can be one of the following values: <ul style="list-style-type: none"> – TIM_OPmode_SINGLE: Only one pulse will be generated. – TIM_OPmode_REPEATITIVE: Repetitive pulses wil be generated.
Return values	• HAL: status

HAL_TIM_OnePulse_DeInit

Function name	HAL_StatusTypeDef HAL_TIM_OnePulse_DeInit (TIM_HandleTypeDef * htim)
Function description	DeInitializes the TIM One Pulse.
Parameters	• htim: TIM One Pulse handle
Return values	• HAL: status

HAL_TIM_OnePulse_MspInit

Function name	void HAL_TIM_OnePulse_MspInit (TIM_HandleTypeDef * htim)
Function description	Initializes the TIM One Pulse MSP.
Parameters	• htim: TIM handle
Return values	• None:

HAL_TIM_OnePulse_MspDeInit

Function name	void HAL_TIM_OnePulse_MspDeInit (TIM_HandleTypeDef * htim)
Function description	DeInitializes TIM One Pulse MSP.
Parameters	• htim: TIM handle
Return values	• None:

HAL_TIM_OnePulse_Start

Function name	HAL_StatusTypeDef HAL_TIM_OnePulse_Start (TIM_HandleTypeDef * htim, uint32_t OutputChannel)
---------------	--

Function description	Starts the TIM One Pulse signal generation.
Parameters	<ul style="list-style-type: none"> • htim: : TIM One Pulse handle • OutputChannel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_OnePulse_Stop

Function name	HAL_StatusTypeDef HAL_TIM_OnePulse_Stop (TIM_HandleTypeDef * htim, uint32_t OutputChannel)
Function description	Stops the TIM One Pulse signal generation.
Parameters	<ul style="list-style-type: none"> • htim: : TIM One Pulse handle • OutputChannel: : TIM Channels to be disable This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_OnePulse_Start_IT

Function name	HAL_StatusTypeDef HAL_TIM_OnePulse_Start_IT (TIM_HandleTypeDef * htim, uint32_t OutputChannel)
Function description	Starts the TIM One Pulse signal generation in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM One Pulse handle • OutputChannel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_OnePulse_Stop_IT

Function name	HAL_StatusTypeDef HAL_TIM_OnePulse_Stop_IT (TIM_HandleTypeDef * htim, uint32_t OutputChannel)
Function description	Stops the TIM One Pulse signal generation in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM One Pulse handle • OutputChannel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Encoder_Init

Function name	HAL_StatusTypeDef HAL_TIM_Encoder_Init (TIM_HandleTypeDef * htim, TIM_Encoder_InitTypeDef *
---------------	--

sConfig)

Function description	Initializes the TIM Encoder Interface and create the associated handle.
Parameters	<ul style="list-style-type: none"> • htim: TIM Encoder Interface handle • sConfig: TIM Encoder Interface configuration structure
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Encoder_DelInit

Function name	HAL_StatusTypeDef HAL_TIM_Encoder_DelInit (TIM_HandleTypeDef * htim)
Function description	Deinitializes the TIM Encoder interface.
Parameters	<ul style="list-style-type: none"> • htim: TIM Encoder handle
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Encoder_MspInit

Function name	void HAL_TIM_Encoder_MspInit (TIM_HandleTypeDef * htim)
Function description	Initializes the TIM Encoder Interface MSP.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_Encoder_MspDelInit

Function name	void HAL_TIM_Encoder_MspDelInit (TIM_HandleTypeDef * htim)
Function description	Deinitializes TIM Encoder Interface MSP.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_Encoder_Start

Function name	HAL_StatusTypeDef HAL_TIM_Encoder_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Starts the TIM Encoder Interface.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Encoder Interface handle • Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_ALL: TIM Channel 1 and TIM Channel 2 are selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Encoder_Stop

Function name	HAL_StatusTypeDef HAL_TIM_Encoder_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the TIM Encoder Interface.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Encoder Interface handle • Channel: : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_ALL: TIM Channel 1 and TIM Channel 2 are selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Encoder_Start_IT

Function name	HAL_StatusTypeDef HAL_TIM_Encoder_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Starts the TIM Encoder Interface in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Encoder Interface handle • Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_ALL: TIM Channel 1 and TIM Channel 2 are selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Encoder_Stop_IT

Function name	HAL_StatusTypeDef HAL_TIM_Encoder_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the TIM Encoder Interface in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM Encoder Interface handle • Channel: : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_ALL: TIM Channel 1 and TIM Channel 2 are selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_Encoder_Start_DMA

Function name	HAL_StatusTypeDef HAL_TIM_Encoder_Start_DMA (TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData1, uint32_t * pData2, uint16_t Length)
Function description	Starts the TIM Encoder Interface in DMA mode.

Parameters	<ul style="list-style-type: none"> htim: : TIM Encoder Interface handle Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_ALL: TIM Channel 1 and TIM Channel 2 are selected pData1: The destination Buffer address for IC1. pData2: The destination Buffer address for IC2. Length: The length of data to be transferred from TIM peripheral to memory.
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_Encoder_Stop_DMA

Function name	HAL_StatusTypeDef HAL_TIM_Encoder_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Stops the TIM Encoder Interface in DMA mode.
Parameters	<ul style="list-style-type: none"> htim: : TIM Encoder Interface handle Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_ALL: TIM Channel 1 and TIM Channel 2 are selected
Return values	<ul style="list-style-type: none"> HAL: status

HAL_TIM_IRQHandler

Function name	void HAL_TIM_IRQHandler (TIM_HandleTypeDef * htim)
Function description	This function handles TIM interrupts requests.
Parameters	<ul style="list-style-type: none"> htim: TIM handle
Return values	<ul style="list-style-type: none"> None:

HAL_TIM_OC_ConfigChannel

Function name	HAL_StatusTypeDef HAL_TIM_OC_ConfigChannel (TIM_HandleTypeDef * htim, TIM_OC_InitTypeDef * sConfig, uint32_t Channel)
Function description	Initializes the TIM Output Compare Channels according to the specified parameters in the TIM_OC_InitTypeDef.
Parameters	<ul style="list-style-type: none"> htim: TIM Output Compare handle sConfig: TIM Output Compare configuration structure Channel: : TIM Channels to configure This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected

- TIM_CHANNEL_4: TIM Channel 4 selected
- **HAL:** status

HAL_TIM_PWM_ConfigChannel

Function name	HAL_StatusTypeDef HAL_TIM_PWM_ConfigChannel (TIM_HandleTypeDef * htim, TIM_OC_InitTypeDef * sConfig, uint32_t Channel)
Function description	Initializes the TIM PWM channels according to the specified parameters in the TIM_OC_InitTypeDef.
Parameters	<ul style="list-style-type: none"> • htim: TIM PWM handle • sConfig: TIM PWM configuration structure • Channel: : TIM Channels to be configured This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_IC_ConfigChannel

Function name	HAL_StatusTypeDef HAL_TIM_IC_ConfigChannel (TIM_HandleTypeDef * htim, TIM_IC_InitTypeDef * sConfig, uint32_t Channel)
Function description	Initializes the TIM Input Capture Channels according to the specified parameters in the TIM_IC_InitTypeDef.
Parameters	<ul style="list-style-type: none"> • htim: TIM IC handle • sConfig: TIM Input Capture configuration structure • Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_OnePulse_ConfigChannel

Function name	HAL_StatusTypeDef HAL_TIM_OnePulse_ConfigChannel (TIM_HandleTypeDef * htim, TIM_OnePulse_InitTypeDef * sConfig, uint32_t OutputChannel, uint32_t InputChannel)
Function description	Initializes the TIM One Pulse Channels according to the specified parameters in the TIM_OnePulse_InitTypeDef.
Parameters	<ul style="list-style-type: none"> • htim: TIM One Pulse handle • sConfig: TIM One Pulse configuration structure • OutputChannel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected

- **TIM_CHANNEL_2:** TIM Channel 2 selected
 - **InputChannel:** : TIM Channels to be enabled This parameter can be one of the following values:
 - **TIM_CHANNEL_1:** TIM Channel 1 selected
 - **TIM_CHANNEL_2:** TIM Channel 2 selected
- Return values**
- **HAL:** status

HAL_TIM_ConfigOCrefClear

Function name	HAL_StatusTypeDef HAL_TIM_ConfigOCrefClear (TIM_HandleTypeDef * htim, TIM_ClearInputConfigTypeDef * sClearInputConfig, uint32_t Channel)
Function description	Configures the OCRef clear feature.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle • sClearInputConfig: pointer to a TIM_ClearInputConfigTypeDef structure that contains the OCREF clear feature and parameters for the TIM peripheral. • Channel: specifies the TIM Channel This parameter can be one of the following values: <ul style="list-style-type: none"> – TIM_CHANNEL_1: TIM Channel 1 – TIM_CHANNEL_2: TIM Channel 2 – TIM_CHANNEL_3: TIM Channel 3 – TIM_CHANNEL_4: TIM Channel 4
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_ConfigClockSource

Function name	HAL_StatusTypeDef HAL_TIM_ConfigClockSource (TIM_HandleTypeDef * htim, TIM_ClockConfigTypeDef * sClockSourceConfig)
Function description	Configures the clock source to be used.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle • sClockSourceConfig: pointer to a TIM_ClockConfigTypeDef structure that contains the clock source information for the TIM peripheral.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_ConfigTI1Input

Function name	HAL_StatusTypeDef HAL_TIM_ConfigTI1Input (TIM_HandleTypeDef * htim, uint32_t TI1_Selection)
Function description	Selects the signal connected to the TI1 input: direct from CH1_input or a XOR combination between CH1_input, CH2_input & CH3_input.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle. • TI1_Selection: Indicate whether or not channel 1 is connected to the output of a XOR gate. This parameter can be one of the following values: <ul style="list-style-type: none"> – TIM_TI1SELECTION_CH1: The TIMx_CH1 pin is

- connected to TI1 input
- **TIM_TI1SELECTION_XORCOMBINATION:** The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Return values

- **HAL:** status

HAL_TIM_SlaveConfigSynchronization

Function name	HAL_StatusTypeDef HAL_TIM_SlaveConfigSynchronization (TIM_HandleTypeDef * htim, TIM_SlaveConfigTypeDef * sSlaveConfig)
Function description	Configures the TIM in Slave mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle. • sSlaveConfig: pointer to a TIM_SlaveConfigTypeDef structure that contains the selected trigger (internal trigger input, filtered timer input or external trigger input) and the) and the Slave mode (Disable, Reset, Gated, Trigger, External clock mode 1).
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_SlaveConfigSynchronization_IT

Function name	HAL_StatusTypeDef HAL_TIM_SlaveConfigSynchronization_IT (TIM_HandleTypeDef * htim, TIM_SlaveConfigTypeDef * sSlaveConfig)
Function description	Configures the TIM in Slave mode in interrupt mode.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle. • sSlaveConfig: pointer to a TIM_SlaveConfigTypeDef structure that contains the selected trigger (internal trigger input, filtered timer input or external trigger input) and the) and the Slave mode (Disable, Reset, Gated, Trigger, External clock mode 1).
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_TIM_DMABurst_WriteStart

Function name	HAL_StatusTypeDef HAL_TIM_DMABurst_WriteStart (TIM_HandleTypeDef * htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t * BurstBuffer, uint32_t BurstLength)
Function description	Configure the DMA Burst to transfer Data from the memory to the TIM peripheral.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle • BurstBaseAddress: : TIM Base address from where the DMA will start the Data write This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_DMABASE_CR1 - TIM_DMABASE_CR2 - TIM_DMABASE_SMCR

- TIM_DMABASE_DIER
 - TIM_DMABASE_SR
 - TIM_DMABASE_EGR
 - TIM_DMABASE_CCMR1
 - TIM_DMABASE_CCMR2
 - TIM_DMABASE_CCER
 - TIM_DMABASE_CNT
 - TIM_DMABASE_PSC
 - TIM_DMABASE_ARR
 - TIM_DMABASE_CCR1
 - TIM_DMABASE_CCR2
 - TIM_DMABASE_CCR3
 - TIM_DMABASE_CCR4
 - TIM_DMABASE_DCR
 - **BurstRequestSrc:** TIM DMA Request sources This parameter can be one of the following values:
 - TIM_DMA_UPDATE: TIM update Interrupt source
 - TIM_DMA_CC1: TIM Capture Compare 1 DMA source
 - TIM_DMA_CC2: TIM Capture Compare 2 DMA source
 - TIM_DMA_CC3: TIM Capture Compare 3 DMA source
 - TIM_DMA_CC4: TIM Capture Compare 4 DMA source
 - TIM_DMA_TRIGGER: TIM Trigger DMA source
 - **BurstBuffer:** The Buffer address.
 - **BurstLength:** DMA Burst length. This parameter can be one value between: TIM_DMABURSTLENGTH_1TRANSFER and TIM_DMABURSTLENGTH_18TRANSFERS.
- Return values**
- **HAL:** status

HAL_TIM_DMABurst_WriteStop

Function name	HAL_StatusTypeDef HAL_TIM_DMABurst_WriteStop (TIM_HandleTypeDef * htim, uint32_t BurstRequestSrc)
Function description	Stops the TIM DMA Burst mode.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle • BurstRequestSrc: TIM DMA Request sources to disable

Return values

- **HAL:** status

HAL_TIM_DMABurst_ReadStart

Function name	HAL_StatusTypeDef HAL_TIM_DMABurst_ReadStart (TIM_HandleTypeDef * htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t * BurstBuffer, uint32_t BurstLength)
Function description	Configure the DMA Burst to transfer Data from the TIM peripheral to the memory.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle • BurstBaseAddress: : TIM Base address from where the DMA will starts the Data read This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_DMABASE_CR1

- TIM_DMABASE_CR2
 - TIM_DMABASE_SMCR
 - TIM_DMABASE_DIER
 - TIM_DMABASE_SR
 - TIM_DMABASE_EGR
 - TIM_DMABASE_CCMR1
 - TIM_DMABASE_CCMR2
 - TIM_DMABASE_CCER
 - TIM_DMABASE_CNT
 - TIM_DMABASE_PSC
 - TIM_DMABASE_ARR
 - TIM_DMABASE_CCR1
 - TIM_DMABASE_CCR2
 - TIM_DMABASE_CCR3
 - TIM_DMABASE_CCR4
 - TIM_DMABASE_DCR
 - **BurstRequestSrc:** TIM DMA Request sources This parameter can be one of the following values:
 - TIM_DMA_UPDATE: TIM update Interrupt source
 - TIM_DMA_CC1: TIM Capture Compare 1 DMA source
 - TIM_DMA_CC2: TIM Capture Compare 2 DMA source
 - TIM_DMA_CC3: TIM Capture Compare 3 DMA source
 - TIM_DMA_CC4: TIM Capture Compare 4 DMA source
 - TIM_DMA_TRIGGER: TIM Trigger DMA source
 - **BurstBuffer:** The Buffer address.
 - **BurstLength:** DMA Burst length. This parameter can be one value between: TIM_DMABURSTLENGTH_1TRANSFER and TIM_DMABURSTLENGTH_18TRANSFERS.
- Return values**
- **HAL:** status

HAL_TIM_DMABurst_ReadStop

Function name	HAL_StatusTypeDef HAL_DMABurst_ReadStop (TIM_HandleTypeDef * htim, uint32_t BurstRequestSrc)
Function description	Stop the DMA burst reading.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle • BurstRequestSrc: TIM DMA Request sources to disable.
Return values	• HAL: status

HAL_TIM_GenerateEvent

Function name	HAL_StatusTypeDef HAL_TIM_GenerateEvent (TIM_HandleTypeDef * htim, uint32_t EventSource)
Function description	Generate a software event.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle • EventSource: specifies the event source. This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_EVENTSOURCE_UPDATE: Timer update Event source - TIM_EVENTSOURCE_CC1: Timer Capture Compare 1

	<ul style="list-style-type: none"> - Event source - TIM_EVENTSOURCE_CC2: Timer Capture Compare 2 Event source - TIM_EVENTSOURCE_CC3: Timer Capture Compare 3 Event source - TIM_EVENTSOURCE_CC4: Timer Capture Compare 4 Event source - TIM_EVENTSOURCE_TRIGGER: Timer Trigger Event source
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • TIM6 and TIM7 can only generate an update event.

HAL_TIM_ReadCapturedValue

Function name	uint32_t HAL_TIM_ReadCapturedValue (TIM_HandleTypeDef * htim, uint32_t Channel)
Function description	Read the captured value from Capture Compare unit.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle. • Channel: : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> - TIM_CHANNEL_1: TIM Channel 1 selected - TIM_CHANNEL_2: TIM Channel 2 selected - TIM_CHANNEL_3: TIM Channel 3 selected - TIM_CHANNEL_4: TIM Channel 4 selected
Return values	<ul style="list-style-type: none"> • Captured: value

HAL_TIM_PeriodElapsedCallback

Function name	void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)
Function description	Period elapsed callback in non blocking mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_OC_DelayElapsedCallback

Function name	void HAL_TIM_OC_DelayElapsedCallback (TIM_HandleTypeDef * htim)
Function description	Output Compare callback in non blocking mode.
Parameters	<ul style="list-style-type: none"> • htim: : TIM OC handle
Return values	<ul style="list-style-type: none"> • None:

HAL_TIM_IC_CaptureCallback

Function name	void HAL_TIM_IC_CaptureCallback (TIM_HandleTypeDef * htim)
Function description	Input Capture callback in non blocking mode.

Parameters	<ul style="list-style-type: none">• htim: : TIM IC handle
Return values	<ul style="list-style-type: none">• None:

HAL_TIM_PWM_PulseFinishedCallback

Function name	void HAL_TIM_PWM_PulseFinishedCallback (TIM_HandleTypeDef * htim)
Function description	PWM Pulse finished callback in non blocking mode.
Parameters	<ul style="list-style-type: none">• htim: : TIM handle
Return values	<ul style="list-style-type: none">• None:

HAL_TIM_TriggerCallback

Function name	void HAL_TIM_TriggerCallback (TIM_HandleTypeDef * htim)
Function description	Hall Trigger detection callback in non blocking mode.
Parameters	<ul style="list-style-type: none">• htim: : TIM handle
Return values	<ul style="list-style-type: none">• None:

HAL_TIM_ErrorCallback

Function name	void HAL_TIM_ErrorCallback (TIM_HandleTypeDef * htim)
Function description	Timer error callback in non blocking mode.
Parameters	<ul style="list-style-type: none">• htim: : TIM handle
Return values	<ul style="list-style-type: none">• None:

HAL_TIM_Base_GetState

Function name	HAL_TIM_StateTypeDef HAL_TIM_Base_GetState (TIM_HandleTypeDef * htim)
Function description	Return the TIM Base state.
Parameters	<ul style="list-style-type: none">• htim: TIM Base handle
Return values	<ul style="list-style-type: none">• HAL: state

HAL_TIM_OC_GetState

Function name	HAL_TIM_StateTypeDef HAL_TIM_OC_GetState (TIM_HandleTypeDef * htim)
Function description	Return the TIM OC state.
Parameters	<ul style="list-style-type: none">• htim: TIM Ouput Compare handle
Return values	<ul style="list-style-type: none">• HAL: state

HAL_TIM_PWM_GetState

Function name	HAL_TIM_StateTypeDef HAL_TIM_PWM_GetState (TIM_HandleTypeDef * htim)
Function description	Return the TIM PWM state.

Function description	Return the TIM PWM state.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_TIM_IC_GetState

Function name	HAL_TIM_StateTypeDef HAL_TIM_IC_GetState (TIM_HandleTypeDef * htim)
Function description	Return the TIM Input Capture state.
Parameters	<ul style="list-style-type: none"> • htim: TIM IC handle
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_TIM_OnePulse_GetState

Function name	HAL_TIM_StateTypeDef HAL_TIM_OnePulse_GetState (TIM_HandleTypeDef * htim)
Function description	Return the TIM One Pulse Mode state.
Parameters	<ul style="list-style-type: none"> • htim: TIM OPM handle
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_TIM_Encoder_GetState

Function name	HAL_TIM_StateTypeDef HAL_TIM_Encoder_GetState (TIM_HandleTypeDef * htim)
Function description	Return the TIM Encoder Mode state.
Parameters	<ul style="list-style-type: none"> • htim: TIM Encoder handle
Return values	<ul style="list-style-type: none"> • HAL: state

TIM_DMADelayPulseCplt

Function name	void TIM_DMADelayPulseCplt (DMA_HandleTypeDef * hdma)
Function description	TIM DMA Delay Pulse complete callback.
Parameters	<ul style="list-style-type: none"> • hdma: : pointer to DMA handle.
Return values	<ul style="list-style-type: none"> • None:

TIM_DMAError

Function name	void TIM_DMAError (DMA_HandleTypeDef * hdma)
Function description	TIM DMA error callback.
Parameters	<ul style="list-style-type: none"> • hdma: : pointer to DMA handle.
Return values	<ul style="list-style-type: none"> • None:

TIM_DMACaptureCplt

Function name	void TIM_DMACaptureCplt (DMA_HandleTypeDef * hdma)
Function description	TIM DMA Capture complete callback.
Parameters	<ul style="list-style-type: none">• hdma: : pointer to DMA handle.
Return values	<ul style="list-style-type: none">• None:

43.3 TIM Firmware driver defines

43.3.1 TIM

TIM Automatic Output Enable

TIM_AUTOMATICOUTPUT_ENABLE

TIM_AUTOMATICOUTPUT_DISABLE

TIM Channel

TIM_CHANNEL_1

TIM_CHANNEL_2

TIM_CHANNEL_3

TIM_CHANNEL_4

TIM_CHANNEL_ALL

TIM Capture/Compare Channel State

TIM_CCx_ENABLE

TIM_CCx_DISABLE

TIM ClearInput Polarity

TIM_CLEARINPUTPOLARITY_INVERTED Polarity for ETRx pin

TIM_CLEARINPUTPOLARITY_NONINVERTED Polarity for ETRx pin

TIM ClearInput Prescaler

TIM_CLEARINPUTPRESCALER_DIV1 No prescaler is used

TIM_CLEARINPUTPRESCALER_DIV2 Prescaler for External ETR pin: Capture performed once every 2 events.

TIM_CLEARINPUTPRESCALER_DIV4 Prescaler for External ETR pin: Capture performed once every 4 events.

TIM_CLEARINPUTPRESCALER_DIV8 Prescaler for External ETR pin: Capture performed once every 8 events.

TIM ClearInput Source

TIM_CLEARINPUTSOURCE_ETR

TIM_CLEARINPUTSOURCE_OCREFCLR

TIM_CLEARINPUTSOURCE_NONE

TIM ClockDivision

TIM_CLOCKDIVISION_DIV1

`TIM_CLOCKDIVISION_DIV2`

`TIM_CLOCKDIVISION_DIV4`

TIM Clock Polarity

`TIM_CLOCKPOLARITY_INVERTED` Polarity for ETRx clock sources

`TIM_CLOCKPOLARITY_NONINVERTED` Polarity for ETRx clock sources

`TIM_CLOCKPOLARITY_RISING` Polarity for TIx clock sources

`TIM_CLOCKPOLARITY_FALLING` Polarity for TIx clock sources

`TIM_CLOCKPOLARITY_BOTHEDGE` Polarity for TIx clock sources

TIM Clock Prescaler

`TIM_CLOCKPRESCALER_DIV1` No prescaler is used

`TIM_CLOCKPRESCALER_DIV2` Prescaler for External ETR Clock: Capture performed once every 2 events.

`TIM_CLOCKPRESCALER_DIV4` Prescaler for External ETR Clock: Capture performed once every 4 events.

`TIM_CLOCKPRESCALER_DIV8` Prescaler for External ETR Clock: Capture performed once every 8 events.

TIM Clock Source

`TIM_CLOCKSOURCE_ETRMODE2`

`TIM_CLOCKSOURCE_INTERNAL`

`TIM_CLOCKSOURCE_ITR0`

`TIM_CLOCKSOURCE_ITR1`

`TIM_CLOCKSOURCE_ITR2`

`TIM_CLOCKSOURCE_ITR3`

`TIM_CLOCKSOURCE_TI1ED`

`TIM_CLOCKSOURCE_TI1`

`TIM_CLOCKSOURCE_TI2`

`TIM_CLOCKSOURCE_ETRMODE1`

TIM Counter Mode

`TIM_COUNTERMODE_UP`

`TIM_COUNTERMODE_DOWN`

`TIM_COUNTERMODE_CENTERALIGNED1`

`TIM_COUNTERMODE_CENTERALIGNED2`

`TIM_COUNTERMODE_CENTERALIGNED3`

TIM DMA Base Address

`TIM_DMABASE_CR1`

`TIM_DMABASE_CR2`

`TIM_DMABASE_SMCR`

TIM_DMABASE_DIER
TIM_DMABASE_SR
TIM_DMABASE_EGR
TIM_DMABASE_CCMR1
TIM_DMABASE_CCMR2
TIM_DMABASE_CCER
TIM_DMABASE_CNT
TIM_DMABASE_PSC
TIM_DMABASE_ARR
TIM_DMABASE_CCR1
TIM_DMABASE_CCR2
TIM_DMABASE_CCR3
TIM_DMABASE_CCR4
TIM_DMABASE_DCR
TIM_DMABASE_OR

TIM DMA Burst Length

TIM_DMABURSTLENGTH_1TRANSFER
TIM_DMABURSTLENGTH_2TRANSFERS
TIM_DMABURSTLENGTH_3TRANSFERS
TIM_DMABURSTLENGTH_4TRANSFERS
TIM_DMABURSTLENGTH_5TRANSFERS
TIM_DMABURSTLENGTH_6TRANSFERS
TIM_DMABURSTLENGTH_7TRANSFERS
TIM_DMABURSTLENGTH_8TRANSFERS
TIM_DMABURSTLENGTH_9TRANSFERS
TIM_DMABURSTLENGTH_10TRANSFERS
TIM_DMABURSTLENGTH_11TRANSFERS
TIM_DMABURSTLENGTH_12TRANSFERS
TIM_DMABURSTLENGTH_13TRANSFERS
TIM_DMABURSTLENGTH_14TRANSFERS
TIM_DMABURSTLENGTH_15TRANSFERS
TIM_DMABURSTLENGTH_16TRANSFERS
TIM_DMABURSTLENGTH_17TRANSFERS
TIM_DMABURSTLENGTH_18TRANSFERS

TIM DMA Handle Index

TIM_DMA_ID_UPDATE Index of the DMA handle used for Update DMA requests

<code>TIM_DMA_ID_CC1</code>	Index of the DMA handle used for Capture/Compare 1 DMA requests
<code>TIM_DMA_ID_CC2</code>	Index of the DMA handle used for Capture/Compare 2 DMA requests
<code>TIM_DMA_ID_CC3</code>	Index of the DMA handle used for Capture/Compare 3 DMA requests
<code>TIM_DMA_ID_CC4</code>	Index of the DMA handle used for Capture/Compare 4 DMA requests
<code>TIM_DMA_ID_TRIGGER</code>	Index of the DMA handle used for Trigger DMA requests
<i>TIM DMA Sources</i>	
<code>TIM_DMA_UPDATE</code>	
<code>TIM_DMA_CC1</code>	
<code>TIM_DMA_CC2</code>	
<code>TIM_DMA_CC3</code>	
<code>TIM_DMA_CC4</code>	
<code>TIM_DMA_TRIGGER</code>	
<i>TIM Encoder Mode</i>	
<code>TIM_ENCODERMODE_TI1</code>	
<code>TIM_ENCODERMODE_TI2</code>	
<code>TIM_ENCODERMODE_TI12</code>	
<i>TIM ETR Polarity</i>	
<code>TIM_ETRPOLARITY_INVERTED</code>	Polarity for ETR source
<code>TIM_ETRPOLARITY_NONINVERTED</code>	Polarity for ETR source
<i>TIM ETR Prescaler</i>	
<code>TIM_ETRPRESCALER_DIV1</code>	No prescaler is used
<code>TIM_ETRPRESCALER_DIV2</code>	ETR input source is divided by 2
<code>TIM_ETRPRESCALER_DIV4</code>	ETR input source is divided by 4
<code>TIM_ETRPRESCALER_DIV8</code>	ETR input source is divided by 8
<i>TIM Event Source</i>	
<code>TIM_EVENTSOURCE_UPDATE</code>	
<code>TIM_EVENTSOURCE_CC1</code>	
<code>TIM_EVENTSOURCE_CC2</code>	
<code>TIM_EVENTSOURCE_CC3</code>	
<code>TIM_EVENTSOURCE_CC4</code>	
<code>TIM_EVENTSOURCE_TRIGGER</code>	
<i>TIM Exported Macros</i>	
<code>__HAL_TIM_RESET_HANDLE_STATE</code>	Description:
<code>ATE</code>	<ul style="list-style-type: none"> • Reset TIM handle state.

Parameters:

- `_HANDLE_`: TIM handle.

Return value:

- None

`_HAL_TIM_ENABLE`

Description:

- Enable the TIM peripheral.

Parameters:

- `_HANDLE_`: TIM handle

Return value:

- None

`_HAL_TIM_DISABLE`

Description:

- Disable the TIM peripheral.

Parameters:

- `_HANDLE_`: TIM handle

Return value:

- None

`_HAL_TIM_ENABLE_IT`

Description:

- Enables the specified TIM interrupt.

Parameters:

- `_HANDLE_`: specifies the TIM Handle.
- `_INTERRUPT_`: specifies the TIM interrupt source to enable. This parameter can be one of the following values:
 - `TIM_IT_UPDATE`: Update interrupt
 - `TIM_IT_CC1`: Capture/Compare 1 interrupt
 - `TIM_IT_CC2`: Capture/Compare 2 interrupt
 - `TIM_IT_CC3`: Capture/Compare 3 interrupt
 - `TIM_IT_CC4`: Capture/Compare 4 interrupt
 - `TIM_IT_COM`: Commutation interrupt
 - `TIM_IT_TRIGGER`: Trigger interrupt

Return value:

- None

`_HAL_TIM_DISABLE_IT`

Description:

- Disables the specified TIM interrupt.

Parameters:

- `_HANDLE_`: specifies the TIM Handle.
- `_INTERRUPT_`: specifies the TIM interrupt source to disable. This parameter can be one of the following values:
 - `TIM_IT_UPDATE`: Update interrupt
 - `TIM_IT_CC1`: Capture/Compare 1 interrupt

- TIM_IT_CC2: Capture/Compare 2 interrupt
- TIM_IT_CC3: Capture/Compare 3 interrupt
- TIM_IT_CC4: Capture/Compare 4 interrupt
- TIM_IT_COM: Commutation interrupt
- TIM_IT_TRIGGER: Trigger interrupt

Return value:

- None

`_HAL_TIM_ENABLE_DMA`

Description:

- Enables the specified DMA request.

Parameters:

- `_HANDLE_`: specifies the TIM Handle.
- `_DMA_`: specifies the TIM DMA request to enable. This parameter can be one of the following values:
 - TIM_DMA_UPDATE: Update DMA request
 - TIM_DMA_CC1: Capture/Compare 1 DMA request
 - TIM_DMA_CC2: Capture/Compare 2 DMA request
 - TIM_DMA_CC3: Capture/Compare 3 DMA request
 - TIM_DMA_CC4: Capture/Compare 4 DMA request
 - TIM_DMA_COM: Commutation DMA request
 - TIM_DMA_TRIGGER: Trigger DMA request

Return value:

- None

`_HAL_TIM_DISABLE_DMA`

Description:

- Disables the specified DMA request.

Parameters:

- `_HANDLE_`: specifies the TIM Handle.
- `_DMA_`: specifies the TIM DMA request to disable. This parameter can be one of the following values:
 - TIM_DMA_UPDATE: Update DMA request
 - TIM_DMA_CC1: Capture/Compare 1 DMA request
 - TIM_DMA_CC2: Capture/Compare 2 DMA request
 - TIM_DMA_CC3: Capture/Compare 3 DMA request
 - TIM_DMA_CC4: Capture/Compare 4 DMA request
 - TIM_DMA_COM: Commutation DMA request
 - TIM_DMA_TRIGGER: Trigger DMA request

request

Return value:

- None

`__HAL_TIM_GET_FLAG`

Description:

- Checks whether the specified TIM interrupt flag is set or not.

Parameters:

- `__HANDLE__`: specifies the TIM Handle.
- `__FLAG__`: specifies the TIM interrupt flag to check. This parameter can be one of the following values:
 - `TIM_FLAG_UPDATE`: Update interrupt flag
 - `TIM_FLAG_CC1`: Capture/Compare 1 interrupt flag
 - `TIM_FLAG_CC2`: Capture/Compare 2 interrupt flag
 - `TIM_FLAG_CC3`: Capture/Compare 3 interrupt flag
 - `TIM_FLAG_CC4`: Capture/Compare 4 interrupt flag
 - `TIM_FLAG_COM`: Commutation interrupt flag
 - `TIM_FLAG_TRIGGER`: Trigger interrupt flag
 - `TIM_FLAG_CC1OF`: Capture/Compare 1 overcapture flag
 - `TIM_FLAG_CC2OF`: Capture/Compare 2 overcapture flag
 - `TIM_FLAG_CC3OF`: Capture/Compare 3 overcapture flag
 - `TIM_FLAG_CC4OF`: Capture/Compare 4 overcapture flag

Return value:

- The: new state of `__FLAG__` (TRUE or FALSE).

`__HAL_TIM_CLEAR_FLAG`

Description:

- Clears the specified TIM interrupt flag.

Parameters:

- `__HANDLE__`: specifies the TIM Handle.
- `__FLAG__`: specifies the TIM interrupt flag to clear. This parameter can be one of the following values:
 - `TIM_FLAG_UPDATE`: Update interrupt flag
 - `TIM_FLAG_CC1`: Capture/Compare 1 interrupt flag
 - `TIM_FLAG_CC2`: Capture/Compare 2 interrupt flag
 - `TIM_FLAG_CC3`: Capture/Compare 3 interrupt flag

- TIM_FLAG_CC4: Capture/Compare 4 interrupt flag
- TIM_FLAG_COM: Commutation interrupt flag
- TIM_FLAG_TRIGGER: Trigger interrupt flag
- TIM_FLAG_CC1OF: Capture/Compare 1 overcapture flag
- TIM_FLAG_CC2OF: Capture/Compare 2 overcapture flag
- TIM_FLAG_CC3OF: Capture/Compare 3 overcapture flag
- TIM_FLAG_CC4OF: Capture/Compare 4 overcapture flag

Return value:

- The: new state of __FLAG__ (TRUE or FALSE).

[__HAL_TIM_GET_IT_SOURCE](#)**Description:**

- Checks whether the specified TIM interrupt has occurred or not.

Parameters:

- __HANDLE__: TIM handle
- __INTERRUPT__: specifies the TIM interrupt source to check.

Return value:

- The: state of TIM_IT (SET or RESET).

[__HAL_TIM_CLEAR_IT](#)**Description:**

- Clear the TIM interrupt pending bits.

Parameters:

- __HANDLE__: TIM handle
- __INTERRUPT__: specifies the interrupt pending bit to clear.

Return value:

- None

[__HAL_TIM_IS_TIM_COUNTING_DOWN](#)**Description:**

- Indicates whether or not the TIM Counter is used as downcounter.

Parameters:

- __HANDLE__: TIM handle.

Return value:

- False: (Counter used as upcounter) or True (Counter used as downcounter)

Notes:

- This macro is particularly usefull to get the

counting mode when the timer operates in Center-aligned mode or Encoder mode.

__HAL_TIM_SET_PRESCALER

Description:

- Sets the TIM active prescaler register value on update event.

Parameters:

- __HANDLE__: TIM handle.
- __PRESC__: specifies the active prescaler register new value.

Return value:

- None

__HAL_TIM_SET_COMPARE

Description:

- Sets the TIM Capture Compare Register value on runtime without calling another time ConfigChannel function.

Parameters:

- __HANDLE__: TIM handle.
- __CHANNEL__: : TIM Channels to be configured. This parameter can be one of the following values:
 - TIM_CHANNEL_1: TIM Channel 1 selected
 - TIM_CHANNEL_2: TIM Channel 2 selected
 - TIM_CHANNEL_3: TIM Channel 3 selected
 - TIM_CHANNEL_4: TIM Channel 4 selected
- __COMPARE__: specifies the Capture Compare register new value.

Return value:

- None

__HAL_TIM_GET_COMPARE

Description:

- Gets the TIM Capture Compare Register value on runtime.

Parameters:

- __HANDLE__: TIM handle.
- __CHANNEL__: : TIM Channel associated with the capture compare register This parameter can be one of the following values:
 - TIM_CHANNEL_1: get capture/compare 1 register value
 - TIM_CHANNEL_2: get capture/compare 2 register value
 - TIM_CHANNEL_3: get capture/compare 3 register value

- TIM_CHANNEL_4: get capture/compare 4 register value

Return value:

- None

__HAL_TIM_SET_COUNTER

Description:

- Sets the TIM Counter Register value on runtime.

Parameters:

- __HANDLE__: TIM handle.
- __COUNTER__: specifies the Counter register new value.

Return value:

- None

__HAL_TIM_GET_COUNTER

Description:

- Gets the TIM Counter Register value on runtime.

Parameters:

- __HANDLE__: TIM handle.

Return value:

- None

__HAL_TIM_SET_AUTORELOAD

Description:

- Sets the TIM Autoreload Register value on runtime without calling another time any Init function.

Parameters:

- __HANDLE__: TIM handle.
- __AUTORELOAD__: specifies the Counter register new value.

Return value:

- None

__HAL_TIM_GET_AUTORELOAD

Description:

- Gets the TIM Autoreload Register value on runtime.

Parameters:

- __HANDLE__: TIM handle.

Return value:

- None

__HAL_TIM_SET_CLOCKDIVISION

Description:

- Sets the TIM Clock Division value on runtime

without calling another time any Init function.

Parameters:

- `__HANDLE__`: TIM handle.
- `__CKD__`: specifies the clock division value. This parameter can be one of the following value:
 - `TIM_CLOCKDIVISION_DIV1`
 - `TIM_CLOCKDIVISION_DIV2`
 - `TIM_CLOCKDIVISION_DIV4`

Return value:

- None

`__HAL_TIM_GET_CLOCKDIVISION`

N

Description:

- Gets the TIM Clock Division value on runtime.

Parameters:

- `__HANDLE__`: TIM handle.

Return value:

- None

`__HAL_TIM_SET_ICPRESCALER`

Description:

- Sets the TIM Input Capture prescaler on runtime without calling another time

Parameters:

- `__HANDLE__`: TIM handle.
- `__CHANNEL__`: : TIM Channels to be configured. This parameter can be one of the following values:
 - `TIM_CHANNEL_1`: TIM Channel 1 selected
 - `TIM_CHANNEL_2`: TIM Channel 2 selected
 - `TIM_CHANNEL_3`: TIM Channel 3 selected
 - `TIM_CHANNEL_4`: TIM Channel 4 selected
- `__ICPSC__`: specifies the Input Capture4 prescaler new value. This parameter can be one of the following values:
 - `TIM_ICPSC_DIV1`: no prescaler
 - `TIM_ICPSC_DIV2`: capture is done once every 2 events
 - `TIM_ICPSC_DIV4`: capture is done once every 4 events
 - `TIM_ICPSC_DIV8`: capture is done once every 8 events

Return value:

- None

_HAL_TIM_GET_ICPRESCALER **Description:**

- Gets the TIM Input Capture prescaler on runtime.

Parameters:

- HANDLE: TIM handle.
- CHANNEL: : TIM Channels to be configured. This parameter can be one of the following values:
 - TIM_CHANNEL_1: get input capture 1 prescaler value
 - TIM_CHANNEL_2: get input capture 2 prescaler value
 - TIM_CHANNEL_3: get input capture 3 prescaler value
 - TIM_CHANNEL_4: get input capture 4 prescaler value

Return value:

- None

_HAL_TIM_URS_ENABLE**Description:**

- Set the Update Request Source (URS) bit of the TIMx_CR1 register.

Parameters:

- HANDLE: TIM handle.

Return value:

- None

Notes:

- When the URS bit of the TIMx_CR1 register is set, only counter overflow/underflow generates an update interrupt or DMA request (if enabled)

_HAL_TIM_URS_DISABLE**Description:**

- Reset the Update Request Source (URS) bit of the TIMx_CR1 register.

Parameters:

- HANDLE: TIM handle.

Return value:

- None

Notes:

- When the URS bit of the TIMx_CR1 register is reset, any of the following events generate an update interrupt or DMA request (if enabled): (+) Counter overflow/underflow (+) Setting the UG bit (+) Update generation through the slave mode controller

`__HAL_TIM_SET_CAPTUREPOLARITY`**Description:**

- Sets the TIM Capture x input polarity on runtime.

Parameters:

- `__HANDLE__`: TIM handle.
- `__CHANNEL__`: TIM Channels to be configured. This parameter can be one of the following values:
 - `TIM_CHANNEL_1`: TIM Channel 1 selected
 - `TIM_CHANNEL_2`: TIM Channel 2 selected
 - `TIM_CHANNEL_3`: TIM Channel 3 selected
 - `TIM_CHANNEL_4`: TIM Channel 4 selected
- `__POLARITY__`: Polarity for TIx source
 - `TIM_INPUTCHANNELPOLARITY_RISING`: Rising Edge
 - `TIM_INPUTCHANNELPOLARITY_FALLING`: Falling Edge
 - `TIM_INPUTCHANNELPOLARITY_BOTHEDGE`: Rising and Falling Edge

Return value:

- None

Notes:

- The polarity `TIM_INPUTCHANNELPOLARITY_BOTHEDGE` is not authorized for TIM Channel 4.

TIM Flag Definition`TIM_FLAG_UPDATE``TIM_FLAG_CC1``TIM_FLAG_CC2``TIM_FLAG_CC3``TIM_FLAG_CC4``TIM_FLAG_TRIGGER``TIM_FLAG_CC1OF``TIM_FLAG_CC2OF``TIM_FLAG_CC3OF``TIM_FLAG_CC4OF`***TIM Input Capture Polarity***`TIM_ICPOLARITY_RISING``TIM_ICPOLARITY_FALLING`

TIM_ICPOLARITY_BOTHEDGE***TIM Input Capture Prescaler***

TIM_ICPSC_DIV1	Capture performed each time an edge is detected on the capture input
TIM_ICPSC_DIV2	Capture performed once every 2 events
TIM_ICPSC_DIV4	Capture performed once every 4 events
TIM_ICPSC_DIV8	Capture performed once every 8 events

TIM Input Capture Selection

TIM_ICSELECTION_DIRECTTI	TIM Input 1, 2, 3 or 4 is selected to be connected to IC1, IC2, IC3 or IC4, respectively
TIM_ICSELECTION_INDIRECTTI	TIM Input 1, 2, 3 or 4 is selected to be connected to IC2, IC1, IC4 or IC3, respectively
TIM_ICSELECTION_TRC	TIM Input 1, 2, 3 or 4 is selected to be connected to TRC

TIM Input Channel Polarity

TIM_INPUTCHANNELPOLARITY_RISING	Polarity for TIx source
TIM_INPUTCHANNELPOLARITY_FALLING	Polarity for TIx source
TIM_INPUTCHANNELPOLARITY_BOTHEDGE	Polarity for TIx source

TIM Interrupt Definition

TIM_IT_UPDATE
TIM_IT_CC1
TIM_IT_CC2
TIM_IT_CC3
TIM_IT_CC4
TIM_IT_TRIGGER

TIM Lock level

TIM_LOCKLEVEL_OFF
TIM_LOCKLEVEL_1
TIM_LOCKLEVEL_2
TIM_LOCKLEVEL_3

TIM Master Mode Selection

TIM_TRGO_RESET
TIM_TRGO_ENABLE
TIM_TRGO_UPDATE
TIM_TRGO_OC1
TIM_TRGO_OC1REF
TIM_TRGO_OC2REF
TIM_TRGO_OC3REF

TIM_TRGO_OC4REF

TIM Master Slave Mode

TIM_MASTERSLAVEMODE_ENABLE

TIM_MASTERSLAVEMODE_DISABLE

TIM One Pulse Mode

TIM_OPMODE_SINGLE

TIM_OPMODE_REPEATITIVE

TIM OSSI Off State Selection for Idle mode state

TIM_OSSI_ENABLE

TIM_OSSI_DISABLE

TIM OSSR Off State Selection for Run mode state

TIM_OSSR_ENABLE

TIM_OSSR_DISABLE

TIM Output Compare and PWM modes

TIM_OCMODE_TIMING

TIM_OCMODE_ACTIVE

TIM_OCMODE_INACTIVE

TIM_OCMODE_TOGGLE

TIM_OCMODE_PWM1

TIM_OCMODE_PWM2

TIM_OCMODE_FORCED_ACTIVE

TIM_OCMODE_FORCED_INACTIVE

TIM Output Compare Idle State

TIM_OCIDLESTATE_SET

TIM_OCIDLESTATE_RESET

TIM Output Compare Polarity

TIM_OCPOLARITY_HIGH

TIM_OCPOLARITY_LOW

TIM Output Fast State

TIM_OCFAST_DISABLE

TIM_OCFAST_ENABLE

TIM Select

TIM_SELECT_NONE None selected

TIM_SELECT_TIM2 Timer 2 selected

TIM_SELECT_TIM3 Timer 3 selected

TIM_SELECT_TIM4 Timer 4 selected

IS_RI_TIM

TIM Slave Mode

TIM_SLAVERESET_DISABLE
TIM_SLAVERESET_RESET
TIM_SLAVERESET_GATED
TIM_SLAVERESET_TRIGGER
TIM_SLAVERESET_EXTERNAL1

TIM TI1 Input Selection

TIM_TI1SELECTION_CH1
TIM_TI1SELECTION_XORCOMBINATION

TIM Trigger Polarity

TIM_TRIGGERPOLARITY_INVERTED	Polarity for ETRx trigger sources
TIM_TRIGGERPOLARITY_NONINVERTED	Polarity for ETRx trigger sources
TIM_TRIGGERPOLARITY_RISING	Polarity for TIxFPx or TI1_ED trigger sources
TIM_TRIGGERPOLARITY_FALLING	Polarity for TIxFPx or TI1_ED trigger sources
TIM_TRIGGERPOLARITY_BOTHEDGE	Polarity for TIxFPx or TI1_ED trigger sources

TIM Trigger Prescaler

TIM_TRIGGERPRESCALER_DIV1	No prescaler is used
TIM_TRIGGERPRESCALER_DIV2	Prescaler for External ETR Trigger: Capture performed once every 2 events.
TIM_TRIGGERPRESCALER_DIV4	Prescaler for External ETR Trigger: Capture performed once every 4 events.
TIM_TRIGGERPRESCALER_DIV8	Prescaler for External ETR Trigger: Capture performed once every 8 events.

TIM Trigger Selection

TIM_TS_ITR0
TIM_TS_ITR1
TIM_TS_ITR2
TIM_TS_ITR3
TIM_TS_TI1F_ED
TIM_TS_TI1FP1
TIM_TS_TI2FP2
TIM_TS_ETRF
TIM_TS_NONE

44 HAL TIM Extension Driver

44.1 TIMEEx Firmware driver registers structures

44.1.1 TIM_MasterConfigTypeDef

Data Fields

- *uint32_t MasterOutputTrigger*
- *uint32_t MasterSlaveMode*

Field Documentation

- *uint32_t TIM_MasterConfigTypeDef::MasterOutputTrigger*
Trigger output (TRGO) selection This parameter can be a value of
[**TIM_Master_Mode_Selection**](#)
- *uint32_t TIM_MasterConfigTypeDef::MasterSlaveMode*
Master/slave mode selection This parameter can be a value of
[**TIM_Master_Slave_Mode**](#)

44.2 TIMEEx Firmware driver API description

44.2.1 TIMER Extended features

The Timer Extension features include:

1. Synchronization circuit to control the timer with external signals and to interconnect several timers together.
2. Timer remapping capabilities configuration

44.2.2 Peripheral Control functions

This section provides functions allowing to:

- Configure Master synchronization.
- Configure timer remapping capabilities.

This section contains the following APIs:

- [**HAL_TIMEEx_MasterConfigSynchronization\(\)**](#)
- [**HAL_TIMEEx_RemapConfig\(\)**](#)

44.2.3 Detailed description of functions

HAL_TIMEEx_MasterConfigSynchronization

Function name	<code>HAL_StatusTypeDef HAL_TIMEEx_MasterConfigSynchronization (TIM_HandleTypeDef * htim, TIM_MasterConfigTypeDef * sMasterConfig)</code>
Function description	Configures the TIM in master mode.
Parameters	<ul style="list-style-type: none">• htim: TIM handle.• sMasterConfig: pointer to a TIM_MasterConfigTypeDef structure that contains the selected trigger output (TRGO) and the Master/Slave mode.

Return values	<ul style="list-style-type: none"> • HAL: status
HAL_TIMEx_RemapConfig	
Function name	HAL_StatusTypeDef HAL_TIMEx_RemapConfig (TIM_HandleTypeDef *htim, uint32_t Remap)
Function description	Configures the TIM2/TIM3/TIM9/TIM10/TIM11 Remapping input capabilities.
Parameters	<ul style="list-style-type: none"> • htim: TIM handle. • Remap: specifies the TIM remapping source. This parameter is a combination of the following values depending on TIM instance.
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • For TIM2, the parameter can have the following values: TIM_TIM2_ITR1_TIM10_OC: TIM2 ITR1 input is connected to TIM10 OC TIM_TIM2_ITR1_TIM5_TGO: TIM2 ITR1 input is connected to TIM5 TGO • For TIM3, the parameter can have the following values: TIM_TIM3_ITR2_TIM11_OC: TIM3 ITR2 input is connected to TIM11 OC TIM_TIM3_ITR2_TIM5_TGO: TIM3 ITR2 input is connected to TIM5 TGO • For TIM9, the parameter is a combination of 2 fields (field1 field2): <ul style="list-style-type: none"> • For TIM9, the field1 can have the following values: TIM_TIM9_ITR1_TIM3_TGO: TIM9 ITR1 input is connected to TIM3 TGO TIM_TIM9_ITR1_TS: TIM9 ITR1 input is connected to touch sensing I/O • For TIM9, the field2 can have the following values: TIM_TIM9_GPIO: TIM9 Channel1 is connected to GPIO TIM_TIM9_LSE: TIM9 Channel1 is connected to LSE internal clock TIM_TIM9_GPIO1: TIM9 Channel1 is connected to GPIO TIM_TIM9_GPIO2: TIM9 Channel1 is connected to GPIO • For TIM10, the parameter is a combination of 3 fields (field1 field2 field3): <ul style="list-style-type: none"> • For TIM10, the field1 can have the following values: TIM_TIM10_TI1RMP: TIM10 Channel 1 depends on TI1_RMP TIM_TIM10_RI: TIM10 Channel 1 is connected to RI • For TIM10, the field2 can have the following values: TIM_TIM10_ETR_LSE: TIM10 ETR input is connected to LSE clock TIM_TIM10_ETR_TIM9_TGO: TIM10 ETR input is connected to TIM9 TGO • For TIM10, the field3 can have the following values: TIM_TIM10_GPIO: TIM10 Channel1 is connected to GPIO TIM_TIM10_LSI: TIM10 Channel1 is connected to LSI internal clock TIM_TIM10_LSE: TIM10 Channel1 is connected to LSE internal clock TIM_TIM10_RTC: TIM10 Channel1 is connected to RTC wakeup interrupt • For TIM11, the parameter is a combination of 3 fields (field1 field2 field3): <ul style="list-style-type: none"> • For TIM11, the field1 can have the following values:

- TIM_TIM11_TI1RMP: TIM11 Channel 1 depends on TI1_RMP
 TIM_TIM11_RI: TIM11 Channel 1 is connected to RI
- For TIM11, the field2 can have the following values:
 TIM_TIM11_ETR_LSE: TIM11 ETR input is connected to LSE clock
 TIM_TIM11_ETR_TIM9_TGO: TIM11 ETR input is connected to TIM9 TGO
 - For TIM11, the field3 can have the following values:
 TIM_TIM11_GPIO: TIM11 Channel1 is connected to GPIO
 TIM_TIM11_MSI: TIM11 Channel1 is connected to MSI internal clock
 TIM_TIM11_HSE_RTC: TIM11 Channel1 is connected to HSE_RTC clock
 TIM_TIM11_GPIO1: TIM11 Channel1 is connected to GPIO

44.3 TIMEx Firmware driver defines

44.3.1 TIMEx

TIMEx Remap

TIM_TIM2_ITR1_TIM10_OC	TIM2 ITR1 input is connected to TIM10 OC
TIM_TIM2_ITR1_TIM5_TGO	TIM2 ITR1 input is connected to TIM5 TGO
TIM_TIM3_ITR2_TIM11_OC	TIM3 ITR2 input is connected to TIM11 OC
TIM_TIM3_ITR2_TIM5_TGO	TIM3 ITR2 input is connected to TIM5 TGO
TIM_TIM9_ITR1_TIM3_TGO	TIM9 ITR1 input is connected to TIM3 TGO
TIM_TIM9_ITR1_TS	TIM9 ITR1 input is connected to touch sensing I/O
TIM_TIM9_GPIO	TIM9 Channel1 is connected to GPIO
TIM_TIM9_LSE	TIM9 Channel1 is connected to LSE internal clock
TIM_TIM9_GPIO1	TIM9 Channel1 is connected to GPIO
TIM_TIM9_GPIO2	TIM9 Channel1 is connected to GPIO
TIM_TIM10_TI1RMP	TIM10 Channel 1 depends on TI1_RMP
TIM_TIM10_RI	TIM10 Channel 1 is connected to RI
TIM_TIM10_ETR_LSE	TIM10 ETR input is connected to LSE clock
TIM_TIM10_ETR_TIM9_TGO	TIM10 ETR input is connected to TIM9 TGO
TIM_TIM10_GPIO	TIM10 Channel1 is connected to GPIO
TIM_TIM10_LSI	TIM10 Channel1 is connected to LSI internal clock
TIM_TIM10_LSE	TIM10 Channel1 is connected to LSE internal clock
TIM_TIM10_RTC	TIM10 Channel1 is connected to RTC wakeup interrupt
TIM_TIM11_TI1RMP	TIM11 Channel 1 depends on TI1_RMP
TIM_TIM11_RI	TIM11 Channel 1 is connected to RI
TIM_TIM11_ETR_LSE	TIM11 ETR input is connected to LSE clock
TIM_TIM11_ETR_TIM9_TGO	TIM11 ETR input is connected to TIM9 TGO
TIM_TIM11_GPIO	TIM11 Channel1 is connected to GPIO

TIM_TIM11_MSI	TIM11 Channel1 is connected to MSI internal clock
TIM_TIM11_HSE_RTC	TIM11 Channel1 is connected to HSE_RTC clock
TIM_TIM11_GPIO1	TIM11 Channel1 is connected to GPIO
IS_TIM_REMAP	

45 HAL UART Generic Driver

45.1 UART Firmware driver registers structures

45.1.1 UART_InitTypeDef

Data Fields

- *uint32_t BaudRate*
- *uint32_t WordLength*
- *uint32_t StopBits*
- *uint32_t Parity*
- *uint32_t Mode*
- *uint32_t HwFlowCtl*
- *uint32_t OverSampling*

Field Documentation

- ***uint32_t UART_InitTypeDef::BaudRate***

This member configures the UART communication baud rate. The baud rate is computed using the following formula:
 $\text{IntegerDivider} = ((\text{PCLKx}) / (8 * (\text{OVR8}+1) * (\text{uart->Init.BaudRate})))$
 $\text{FractionalDivider} = ((\text{IntegerDivider} - ((\text{uint32_t}) \text{IntegerDivider})) * 8 * (\text{OVR8}+1)) + 0.5$ Where OVR8 is the "oversampling by 8 mode" configuration bit in the CR1 register.

- ***uint32_t UART_InitTypeDef::WordLength***

Specifies the number of data bits transmitted or received in a frame. This parameter can be a value of [**UART_Word_Length**](#)

- ***uint32_t UART_InitTypeDef::StopBits***

Specifies the number of stop bits transmitted. This parameter can be a value of [**UART_Stop_Bits**](#)

- ***uint32_t UART_InitTypeDef::Parity***

Specifies the parity mode. This parameter can be a value of [**UART_Parity**](#)
Note:When parity is enabled, the computed parity is inserted at the MSB position of the transmitted data (9th bit when the word length is set to 9 data bits; 8th bit when the word length is set to 8 data bits).

- ***uint32_t UART_InitTypeDef::Mode***

Specifies whether the Receive or Transmit mode is enabled or disabled. This parameter can be a value of [**UART_Mode**](#)

- ***uint32_t UART_InitTypeDef::HwFlowCtl***

Specifies whether the hardware flow control mode is enabled or disabled. This parameter can be a value of [**UART_Hardware_Flow_Control**](#)

- ***uint32_t UART_InitTypeDef::OverSampling***

Specifies whether the Over sampling 8 is enabled or disabled, to achieve higher speed (up to fPCLK/8). This parameter can be a value of [**UART_Over_Sampling**](#)

45.1.2 UART_HandleTypeDef

Data Fields

- *USART_TypeDef * Instance*
- *UART_InitTypeDef Init*
- *uint8_t * pTxBuffPtr*
- *uint16_t TxXferSize*
- *uint16_t TxXferCount*

- *uint8_t * pRxBuffPtr*
- *uint16_t RxXferSize*
- *uint16_t RxXferCount*
- *DMA_HandleTypeDef * hdmatx*
- *DMA_HandleTypeDef * hdmarx*
- *HAL_LockTypeDef Lock*
- *__IO HAL_UART_StateTypeDef State*
- *__IO uint32_t ErrorCode*

Field Documentation

- ***USART_TypeDef* UART_HandleTypeDef::Instance***
UART registers base address
- ***UART_InitTypeDef UART_HandleTypeDef::Init***
UART communication parameters
- ***uint8_t* UART_HandleTypeDef::pTxBuffPtr***
Pointer to UART Tx transfer Buffer
- ***uint16_t UART_HandleTypeDef::TxXferSize***
UART Tx Transfer size
- ***uint16_t UART_HandleTypeDef::TxXferCount***
UART Tx Transfer Counter
- ***uint8_t* UART_HandleTypeDef::pRxBuffPtr***
Pointer to UART Rx transfer Buffer
- ***uint16_t UART_HandleTypeDef::RxXferSize***
UART Rx Transfer size
- ***uint16_t UART_HandleTypeDef::RxXferCount***
UART Rx Transfer Counter
- ***DMA_HandleTypeDef* UART_HandleTypeDef::hdmatx***
UART Tx DMA Handle parameters
- ***DMA_HandleTypeDef* UART_HandleTypeDef::hdmarx***
UART Rx DMA Handle parameters
- ***HAL_LockTypeDef UART_HandleTypeDef::Lock***
Locking object
- ***__IO HAL_UART_StateTypeDef UART_HandleTypeDef::State***
UART communication state
- ***__IO uint32_t UART_HandleTypeDef::ErrorCode***
UART Error code

45.2 UART Firmware driver API description

45.2.1 How to use this driver

The UART HAL driver can be used as follows:

1. Declare a `UART_HandleTypeDef` handle structure.
2. Initialize the UART low level resources by implementing the `HAL_UART_MspInit()` API:
 - a. Enable the USARTx interface clock.
 - b. UART pins configuration:
 - Enable the clock for the UART GPIOs.
 - Configure the UART pins as alternate function pull-up.
 - c. NVIC configuration if you need to use interrupt process (`HAL_UART_Transmit_IT()` and `HAL_UART_Receive_IT()` APIs):
 - Configure the USARTx interrupt priority.
 - Enable the NVIC USART IRQ handle.

- d. DMA Configuration if you need to use DMA process (HAL_UART_Transmit_DMA() and HAL_UART_Receive_DMA() APIs):
 - Declare a DMA handle structure for the Tx/Rx channel.
 - Enable the DMA interface clock.
 - Configure the declared DMA handle structure with the required Tx/Rx parameters.
 - Configure the DMA Tx/Rx channel.
 - Associate the initialized DMA handle to the UART DMA Tx/Rx handle.
 - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx/Rx channel.
 - Configure the USARTx interrupt priority and enable the NVIC USART IRQ handle (used for last byte sending completion detection in DMA non circular mode)
- 3. Program the Baud Rate, Word Length, Stop Bit, Parity, Hardware flow control and Mode(Receiver/Transmitter) in the uart Init structure.
- 4. For the UART asynchronous mode, initialize the UART registers by calling the HAL_UART_Init() API.
- 5. For the UART Half duplex mode, initialize the UART registers by calling the HAL_HalfDuplex_Init() API.
- 6. For the LIN mode, initialize the UART registers by calling the HAL_LIN_Init() API.
- 7. For the Multi-Processor mode, initialize the UART registers by calling the HAL_MultiProcessor_Init() API.



The specific UART interrupts (Transmission complete interrupt, RXNE interrupt and Error Interrupts) will be managed using the macros __HAL_UART_ENABLE_IT() and __HAL_UART_DISABLE_IT() inside the transmit and receive process.



These APIs (HAL_UART_Init() and HAL_HalfDuplex_Init()) configure also the low level Hardware GPIO, CLOCK, CORTEX...etc) by calling the customized HAL_UART_MspInit() API.

Three operation modes are available within this driver :

Polling mode IO operation

- Send an amount of data in blocking mode using HAL_UART_Transmit()
- Receive an amount of data in blocking mode using HAL_UART_Receive()

Interrupt mode IO operation

- Send an amount of data in non blocking mode using HAL_UART_Transmit_IT()
- At transmission end of transfer HAL_UART_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_UART_TxCpltCallback
- Receive an amount of data in non blocking mode using HAL_UART_Receive_IT()
- At reception end of transfer HAL_UART_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_UART_RxCpltCallback
- In case of transfer Error, HAL_UART_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_UART_ErrorCallback

DMA mode IO operation

- Send an amount of data in non blocking mode (DMA) using HAL_UART_Transmit_DMA()
- At transmission end of half transfer HAL_UART_TxHalfCpltCallback is executed and user can add his own code by customization of function pointer HAL_UART_TxHalfCpltCallback
- At transmission end of transfer HAL_UART_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_UART_TxCpltCallback
- Receive an amount of data in non blocking mode (DMA) using HAL_UART_Receive_DMA()
- At reception end of half transfer HAL_UART_RxHalfCpltCallback is executed and user can add his own code by customization of function pointer HAL_UART_RxHalfCpltCallback
- At reception end of transfer HAL_UART_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_UART_RxCpltCallback
- In case of transfer Error, HAL_UART_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_UART_ErrorCallback
- Pause the DMA Transfer using HAL_UART_DMAPause()
- Resume the DMA Transfer using HAL_UART_DMAResume()
- Stop the DMA Transfer using HAL_UART_DMAStop()

UART HAL driver macros list

Below the list of most used macros in UART HAL driver.

- __HAL_UART_ENABLE: Enable the UART peripheral
- __HAL_UART_DISABLE: Disable the UART peripheral
- __HAL_UART_GET_FLAG : Check whether the specified UART flag is set or not
- __HAL_UART_CLEAR_FLAG : Clear the specified UART pending flag
- __HAL_UART_ENABLE_IT: Enable the specified UART interrupt
- __HAL_UART_DISABLE_IT: Disable the specified UART interrupt
- __HAL_UART_GET_IT_SOURCE: Check whether the specified UART interrupt has occurred or not



You can refer to the UART HAL driver header file for more useful macros

45.2.2 Initialization and Configuration functions

This subsection provides a set of functions allowing to initialize the USARTx or the UARTy in asynchronous mode.

- For the asynchronous mode only these parameters can be configured:
 - Baud Rate
 - Word Length
 - Stop Bit
 - Parity
 - Hardware flow control
 - Receiver/transmitter modes
 - Over Sampling Methode

The HAL_UART_Init(), HAL_HalfDuplex_Init(), HAL_LIN_Init() and HAL_MultiProcessor_Init() APIs follow respectively the UART asynchronous, UART Half

duplex, LIN and Multi-Processor configuration procedures (details for the procedures are available in reference manual (RM0038)).

This section contains the following APIs:

- [*HAL_UART_Init\(\)*](#)
- [*HAL_HalfDuplex_Init\(\)*](#)
- [*HAL_LIN_Init\(\)*](#)
- [*HAL_MultiProcessor_Init\(\)*](#)
- [*HAL_UART_DeInit\(\)*](#)
- [*HAL_UART_MspInit\(\)*](#)
- [*HAL_UART_MspDeInit\(\)*](#)

45.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the UART asynchronous and Half duplex data transfers.

1. There are two modes of transfer:
 - Blocking mode: The communication is performed in polling mode. The HAL status of all data processing is returned by the same function after finishing transfer.
 - Non blocking mode: The communication is performed using Interrupts or DMA, these APIs return the HAL status. The end of the data processing will be indicated through the dedicated UART IRQ when using Interrupt mode or the DMA IRQ when using DMA mode. The *HAL_UART_TxCpltCallback()*, *HAL_UART_RxCpltCallback()* user callbacks will be executed respectively at the end of the transmit or receive process. The *HAL_UART_ErrorCallback()* user callback will be executed when a communication error is detected.
2. Blocking mode APIs are:
 - *HAL_UART_Transmit()*
 - *HAL_UART_Receive()*
3. Non Blocking mode APIs with Interrupt are:
 - *HAL_UART_Transmit_IT()*
 - *HAL_UART_Receive_IT()*
 - *HAL_UART_IRQHandler()*
4. Non Blocking mode functions with DMA are:
 - *HAL_UART_Transmit_DMA()*
 - *HAL_UART_Receive_DMA()*
 - *HAL_UART_DMAPause()*
 - *HAL_UART_DMAResume()*
 - *HAL_UART_DMAStop()*
5. A set of Transfer Complete Callbacks are provided in non blocking mode:
 - *HAL_UART_TxHalfCpltCallback()*
 - *HAL_UART_TxCpltCallback()*
 - *HAL_UART_RxHalfCpltCallback()*
 - *HAL_UART_RxCpltCallback()*
 - *HAL_UART_ErrorCallback()*



In the Half duplex communication, it is forbidden to run the transmit and receive process in parallel, the UART state *HAL_UART_STATE_BUSY_TX_RX* can't be useful.

This section contains the following APIs:

- [*HAL_UART_Transmit\(\)*](#)
- [*HAL_UART_Receive\(\)*](#)
- [*HAL_UART_Transmit_IT\(\)*](#)
- [*HAL_UART_Receive_IT\(\)*](#)
- [*HAL_UART_Transmit_DMA\(\)*](#)
- [*HAL_UART_Receive_DMA\(\)*](#)
- [*HAL_UART_DMAPause\(\)*](#)
- [*HAL_UART_DMAResume\(\)*](#)
- [*HAL_UART_DMAStop\(\)*](#)
- [*HAL_UART_IRQHandler\(\)*](#)
- [*HAL_UART_TxCpltCallback\(\)*](#)
- [*HAL_UART_TxHalfCpltCallback\(\)*](#)
- [*HAL_UART_RxCpltCallback\(\)*](#)
- [*HAL_UART_RxHalfCpltCallback\(\)*](#)
- [*HAL_UART_ErrorCallback\(\)*](#)

45.2.4 Peripheral Control functions

This subsection provides a set of functions allowing to control the UART:

- [*HAL_LIN_SendBreak\(\)*](#) API can be helpful to transmit the break character.
- [*HAL_MultiProcessor_EnterMuteMode\(\)*](#) API can be helpful to enter the UART in mute mode.
- [*HAL_MultiProcessor_ExitMuteMode\(\)*](#) API can be helpful to exit the UART mute mode by software.
- [*HAL_HalfDuplex_EnableTransmitter\(\)*](#) API to enable the UART transmitter and disables the UART receiver in Half Duplex mode
- [*HAL_HalfDuplex_EnableReceiver\(\)*](#) API to enable the UART receiver and disables the UART transmitter in Half Duplex mode

This section contains the following APIs:

- [*HAL_LIN_SendBreak\(\)*](#)
- [*HAL_MultiProcessor_EnterMuteMode\(\)*](#)
- [*HAL_MultiProcessor_ExitMuteMode\(\)*](#)
- [*HAL_HalfDuplex_EnableTransmitter\(\)*](#)
- [*HAL_HalfDuplex_EnableReceiver\(\)*](#)

45.2.5 Peripheral State and Errors functions

This subsection provides a set of functions allowing to return the State of UART communication process, return Peripheral Errors occurred during communication process

- [*HAL_UART_GetState\(\)*](#) API can be helpful to check in run-time the state of the UART peripheral.
- [*HAL_UART_GetError\(\)*](#) check in run-time errors that could be occurred during communication.

This section contains the following APIs:

- [*HAL_UART_GetState\(\)*](#)
- [*HAL_UART_GetError\(\)*](#)

45.2.6 Detailed description of functions

HAL_UART_Init

Function name	HAL_StatusTypeDef HAL_UART_Init (UART_HandleTypeDef * huart)
Function description	Initializes the UART mode according to the specified parameters in the UART_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_HalfDuplex_Init

Function name	HAL_StatusTypeDef HAL_HalfDuplex_Init (UART_HandleTypeDef * huart)
Function description	Initializes the half-duplex mode according to the specified parameters in the UART_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_LIN_Init

Function name	HAL_StatusTypeDef HAL_LIN_Init (UART_HandleTypeDef * huart, uint32_t BreakDetectLength)
Function description	Initializes the LIN mode according to the specified parameters in the UART_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module. • BreakDetectLength: Specifies the LIN break detection length. This parameter can be one of the following values: <ul style="list-style-type: none"> – UART_LINBREAKDETECTLENGTH_10B: 10-bit break detection – UART_LINBREAKDETECTLENGTH_11B: 11-bit break detection
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_MultiProcessor_Init

Function name	HAL_StatusTypeDef HAL_MultiProcessor_Init (UART_HandleTypeDef * huart, uint8_t Address, uint32_t WakeUpMethod)
Function description	Initializes the Multi-Processor mode according to the specified parameters in the UART_InitTypeDef and create the associated

	handle.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module. • Address: UART node address • WakeUpMethod: specifies the UART wakeup method. This parameter can be one of the following values: <ul style="list-style-type: none"> – <code>UART_WAKEUPMETHOD_IDLELINE</code>: Wakeup by an idle line detection – <code>UART_WAKEUPMETHOD_ADDRESSMARK</code>: Wakeup by an address mark
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_UART_DeInit

Function name	<code>HAL_StatusTypeDef HAL_UART_DeInit (UART_HandleTypeDef * huart)</code>
Function description	DeInitializes the UART peripheral.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_UART_MspInit

Function name	<code>void HAL_UART_MspInit (UART_HandleTypeDef * huart)</code>
Function description	UART MSP Init.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • None:

HAL_UART_MspDeInit

Function name	<code>void HAL_UART_MspDeInit (UART_HandleTypeDef * huart)</code>
Function description	UART MSP DeInit.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • None:

HAL_UART_Transmit

Function name	<code>HAL_StatusTypeDef HAL_UART_Transmit (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)</code>
Function description	Sends an amount of data in blocking mode.

Parameters	<ul style="list-style-type: none"> huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module. pData: Pointer to data buffer Size: Amount of data to be sent Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status

HAL_UART_Receive

Function name	HAL_StatusTypeDef HAL_UART_Receive (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)
Function description	Receives an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module. pData: Pointer to data buffer Size: Amount of data to be received Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> HAL: status

HAL_UART_Transmit_IT

Function name	HAL_StatusTypeDef HAL_UART_Transmit_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
Function description	Sends an amount of data in non blocking mode.
Parameters	<ul style="list-style-type: none"> huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module. pData: Pointer to data buffer Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> HAL: status

HAL_UART_Receive_IT

Function name	HAL_StatusTypeDef HAL_UART_Receive_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
Function description	Receives an amount of data in non blocking mode.
Parameters	<ul style="list-style-type: none"> huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module. pData: Pointer to data buffer Size: Amount of data to be received
Return values	<ul style="list-style-type: none"> HAL: status

HAL_UART_Transmit_DMA

Function name	HAL_StatusTypeDef HAL_UART_Transmit_DMA (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
Function description	Sends an amount of data in non blocking mode.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module. • pData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_UART_Receive_DMA

Function name	HAL_StatusTypeDef HAL_UART_Receive_DMA (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
Function description	Receives an amount of data in non blocking mode.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module. • pData: Pointer to data buffer • Size: Amount of data to be received
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • When the UART parity is enabled (PCE = 1), the received data contain the parity bit (MSB position)

HAL_UART_DMAPause

Function name	HAL_StatusTypeDef HAL_UART_DMAPause (UART_HandleTypeDef * huart)
Function description	Pauses the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_UART_DMAResume

Function name	HAL_StatusTypeDef HAL_UART_DMAResume (UART_HandleTypeDef * huart)
Function description	Resumes the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_UART_DMASStop

Function name	HAL_StatusTypeDef HAL_UART_DMASStop (UART_HandleTypeDef * huart)
Function description	Stops the DMA Transfer.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_UART_IRQHandler

Function name	void HAL_UART_IRQHandler (UART_HandleTypeDef * huart)
Function description	This function handles UART interrupt request.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • None:

HAL_UART_TxCpltCallback

Function name	void HAL_UART_TxCpltCallback (UART_HandleTypeDef * huart)
Function description	Tx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • None:

HAL_UART_TxHalfCpltCallback

Function name	void HAL_UART_TxHalfCpltCallback (UART_HandleTypeDef * huart)
Function description	Tx Half Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • None:

HAL_UART_RxCpltCallback

Function name	void HAL_UART_RxCpltCallback (UART_HandleTypeDef * huart)
Function description	Rx Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.

Return values	<ul style="list-style-type: none"> • None:
HAL_UART_RxHalfCpltCallback	
Function name	void HAL_UART_RxHalfCpltCallback (UART_HandleTypeDef * huart)
Function description	Rx Half Transfer completed callbacks.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • None:
HAL_UART_ErrorCallback	
Function name	void HAL_UART_ErrorCallback (UART_HandleTypeDef * huart)
Function description	UART error callbacks.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • None:
HAL_LIN_SendBreak	
Function name	HAL_StatusTypeDef HAL_LIN_SendBreak (UART_HandleTypeDef * huart)
Function description	Transmits break characters.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status
HAL_MultiProcessor_EnterMuteMode	
Function name	HAL_StatusTypeDef HAL_MultiProcessor_EnterMuteMode (UART_HandleTypeDef * huart)
Function description	Enters the UART in mute mode.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status
HAL_MultiProcessor_ExitMuteMode	
Function name	HAL_StatusTypeDef HAL_MultiProcessor_ExitMuteMode (UART_HandleTypeDef * huart)
Function description	Exits the UART mute mode: wake up software.

Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_HalfDuplex_EnableTransmitter

Function name	HAL_StatusTypeDef HAL_HalfDuplex_EnableTransmitter (UART_HandleTypeDef * huart)
Function description	Enables the UART transmitter and disables the UART receiver.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_HalfDuplex_EnableReceiver

Function name	HAL_StatusTypeDef HAL_HalfDuplex_EnableReceiver (UART_HandleTypeDef * huart)
Function description	Enables the UART receiver and disables the UART transmitter.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_UART_GetState

Function name	HAL_UART_StateTypeDef HAL_UART_GetState (UART_HandleTypeDef * huart)
Function description	Returns the UART state.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.
Return values	<ul style="list-style-type: none"> • HAL: state

HAL_UART_GetError

Function name	uint32_t HAL_UART_GetError (UART_HandleTypeDef * huart)
Function description	Return the UART error code.
Parameters	<ul style="list-style-type: none"> • huart: Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART.
Return values	<ul style="list-style-type: none"> • UART: Error Code

45.3 UART Firmware driver defines

45.3.1 UART

UART Error Codes

HAL_UART_ERROR_NONE	No error
HAL_UART_ERROR_PE	Parity error
HAL_UART_ERROR_NE	Noise error
HAL_UART_ERROR_FE	frame error
HAL_UART_ERROR_ORE	Overrun error
HAL_UART_ERROR_DMA	DMA transfer error

UART Exported Macros

`_HAL_UART_RESET_HANDLE_STATE`

Description:

- Reset UART handle state.

Parameters:

- `_HANDLE_`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

`_HAL_UART_FLUSH_DRREGISTER`

Description:

- Flush the UART DR register.

Parameters:

- `_HANDLE_`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

`_HAL_UART_GET_FLAG`

Description:

- Check whether the specified UART flag is set or not.

Parameters:

- `_HANDLE_`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).
- `_FLAG_`: specifies the flag to check. This parameter can be one of the following values:
 - `UART_FLAG_CTS`: CTS Change flag (not available for

- UART4 and UART5)
 - UART_FLAG_LBD: LIN Break detection flag
 - UART_FLAG_TXE: Transmit data register empty flag
 - UART_FLAG_TC: Transmission Complete flag
 - UART_FLAG_RXNE: Receive data register not empty flag
 - UART_FLAG_IDLE: Idle Line detection flag
 - UART_FLAG_ORE: OverRun Error flag
 - UART_FLAG_NE: Noise Error flag
 - UART_FLAG_FE: Framing Error flag
 - UART_FLAG_PE: Parity Error flag

Return value:

- The: new state of __FLAG__ (TRUE or FALSE).

[__HAL_UART_CLEAR_FLAG](#)**Description:**

- Clear the specified UART pending flag.

Parameters:

- __HANDLE__: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).
- __FLAG__: specifies the flag to check. This parameter can be any combination of the following values:
 - UART_FLAG_CTS: CTS Change flag (not available for UART4 and UART5).
 - UART_FLAG_LBD: LIN Break detection flag.
 - UART_FLAG_TC: Transmission Complete flag.
 - UART_FLAG_RXNE: Receive data register not empty flag.

Return value:

- None

Notes:

- PE (Parity error), FE (Framing error), NE (Noise error), ORE (OverRun error) and IDLE (Idle line detected)

flags are cleared by software sequence: a read operation to USART_SR register followed by a read operation to USART_DR register. RXNE flag can be also cleared by a read to the USART_DR register. TC flag can be also cleared by software sequence: a read operation to USART_SR register followed by a write operation to USART_DR register. TXE flag is cleared only by a write to the USART_DR register.

`__HAL_UART_CLEAR_PEFLAG`

Description:

- Clear the UART PE pending flag.

Parameters:

- `__HANDLE__`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

`__HAL_UART_CLEAR_FEFLAG`

Description:

- Clear the UART FE pending flag.

Parameters:

- `__HANDLE__`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

`__HAL_UART_CLEAR_NEFLAG`

Description:

- Clear the UART NE pending flag.

Parameters:

- `__HANDLE__`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

`__HAL_UART_CLEAR_OREFLAG`

Description:

- Clear the UART ORE pending flag.

Parameters:

- `__HANDLE__`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

`__HAL_UART_CLEAR_IDLEFLAG`

Description:

- Clear the UART IDLE pending flag.

Parameters:

- `__HANDLE__`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

`__HAL_UART_ENABLE_IT`

Description:

- Enable the specified UART interrupt.

Parameters:

- `__HANDLE__`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).
- `__INTERRUPT__`: specifies the UART interrupt source to enable. This parameter can be one of the following values:
 - `UART_IT_CTS`: CTS change interrupt
 - `UART_IT_LBD`: LIN Break detection interrupt
 - `UART_IT_TXE`: Transmit Data Register empty interrupt
 - `UART_IT_TC`: Transmission complete interrupt
 - `UART_IT_RXNE`: Receive Data register not empty interrupt
 - `UART_IT_IDLE`: Idle line detection interrupt
 - `UART_IT_PE`: Parity Error interrupt
 - `UART_IT_ERR`: Error interrupt(Frame error, noise)

error, overrun error)

Return value:

- None

_HAL_UART_DISABLE_IT

- Disable the specified UART interrupt.

Parameters:

- HANDLE: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART, UART availability and x,y values depending on device).
- INTERRUPT: specifies the UART interrupt source to disable. This parameter can be one of the following values:
 - UART_IT_CTS: CTS change interrupt
 - UART_IT_LBD: LIN Break detection interrupt
 - UART_IT_TXE: Transmit Data Register empty interrupt
 - UART_IT_TC: Transmission complete interrupt
 - UART_IT_RXNE: Receive Data register not empty interrupt
 - UART_IT_IDLE: Idle line detection interrupt
 - UART_IT_PE: Parity Error interrupt
 - UART_IT_ERR: Error interrupt(Frame error, noise error, overrun error)

Return value:

- None

_HAL_UART_GET_IT_SOURCE

- Check whether the specified UART interrupt has occurred or not.

Parameters:

- HANDLE: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART, UART availability and x,y values depending on device).
- IT: specifies the UART interrupt source to check. This parameter can be one of the following values:
 - UART_IT_CTS: CTS change

- interrupt (not available for UART4 and UART5)
- UART_IT_LBD: LIN Break detection interrupt
- UART_IT_TXE: Transmit Data Register empty interrupt
- UART_IT_TC: Transmission complete interrupt
- UART_IT_RXNE: Receive Data register not empty interrupt
- UART_IT_IDLE: Idle line detection interrupt
- UART_IT_ERR: Error interrupt

Return value:

- The: new state of __IT__ (TRUE or FALSE).

`__HAL_UART_ONE_BIT_SAMPLE_ENABLE`**Description:**

- macros to enables or disables the UART's one bit sampling method

Parameters:

- `__HANDLE__`: specifies the UART Handle. This parameter can be USARTx with x: 1, 2 or 3, or UARTy with y:4 or 5 to select the USART or UART peripheral (availability depending on device for UARTy).

Return value:

- None

`__HAL_UART_ONE_BIT_SAMPLE_DISABLE``__HAL_UART_HWCONTROL_CTS_ENABLE`**Description:**

- Enable CTS flow control This macro allows to enable CTS hardware flow control for a given UART instance, without need to call

Parameters:

- `__HANDLE__`: specifies the UART Handle. This parameter can be any USARTx (supporting the HW Flow control feature). It is used to select the USART peripheral (USART availability and x value depending on device).

Return value:

- None

Notes:

- As macro is expected to be used for

modifying CTS Hw flow control feature activation, without need for USART instance Deinit/Init, following conditions for macro call should be fulfilled : USART instance should have already been initialised (through call of HAL_UART_Init())macro could only be called when corresponding USART instance is disabled (i.e __HAL_UART_DISABLE(__HANDLE__)) and should be followed by an Enable macro (i.e __HAL_UART_ENABLE(__HANDLE__)).

__HAL_UART_HWCONTROL_CTS_DISABLE**Description:**

- Disable CTS flow control This macro allows to disable CTS hardware flow control for a given USART instance, without need to call

Parameters:

- __HANDLE__: specifies the USART Handle. This parameter can be any USARTx (supporting the HW Flow control feature). It is used to select the USART peripheral (USART availability and x value depending on device).

Return value:

- None

Notes:

- As macro is expected to be used for modifying CTS Hw flow control feature activation, without need for USART instance Deinit/Init, following conditions for macro call should be fulfilled : USART instance should have already been initialised (through call of HAL_UART_Init())macro could only be called when corresponding USART instance is disabled (i.e __HAL_UART_DISABLE(__HANDLE__)) and should be followed by an Enable macro (i.e __HAL_UART_ENABLE(__HANDLE__)).

__HAL_UART_HWCONTROL_RTS_ENABLE**Description:**

- Enable RTS flow control This macro allows to enable RTS hardware flow control for a given USART instance, without need to call

Parameters:

- `__HANDLE__`: specifies the UART Handle. This parameter can be any USARTx (supporting the HW Flow control feature). It is used to select the USART peripheral (USART availability and x value depending on device).

Return value:

- None

Notes:

- As macro is expected to be used for modifying RTS Hw flow control feature activation, without need for USART instance Deinit/Init, following conditions for macro call should be fulfilled : USART instance should have already been initialised (through call of `HAL_UART_Init()`) macro could only be called when corresponding USART instance is disabled (i.e `__HAL_UART_DISABLE(__HANDLE__)`) and should be followed by an Enable macro (i.e `__HAL_UART_ENABLE(__HANDLE__)`).

`__HAL_UART_HWCONTROL_RTS_DISABLE`**Description:**

- Disable RTS flow control This macro allows to disable RTS hardware flow control for a given USART instance, without need to call

Parameters:

- `__HANDLE__`: specifies the USART Handle. This parameter can be any USARTx (supporting the HW Flow control feature). It is used to select the USART peripheral (USART availability and x value depending on device).

Return value:

- None

Notes:

- As macro is expected to be used for modifying RTS Hw flow control feature activation, without need for USART instance Deinit/Init, following conditions for macro call should be fulfilled : USART instance should have

already been initialised (through call of HAL_UART_Init())macro could only be called when corresponding UART instance is disabled (i.e __HAL_UART_DISABLE(__HANDLE)) and should be followed by an Enable macro (i.e __HAL_UART_ENABLE(__HANDLE)).

`__HAL_UART_ENABLE`

Description:

- Enable UART.

Parameters:

- `__HANDLE__`: specifies the UART Handle. UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

`__HAL_UART_DISABLE`

Description:

- Disable UART UART Handle selects the USARTx or UARTy peripheral (USART,UART availability and x,y values depending on device).

Return value:

- None

UART Flags

`UART_FLAG_CTS`

`UART_FLAG_LBD`

`UART_FLAG_TXE`

`UART_FLAG_TC`

`UART_FLAG_RXNE`

`UART_FLAG_IDLE`

`UART_FLAG_ORE`

`UART_FLAG_NE`

`UART_FLAG_FE`

`UART_FLAG_PE`

UART Hardware Flow Control

`UART_HWCONTROL_NONE`

`UART_HWCONTROL_RTS`

`UART_HWCONTROL_CTS`

UART_HWCONTROL_RTS_CTS

UART Interrupt Definitions

UART_IT_PE

UART_IT_TXE

UART_IT_TC

UART_IT_RXNE

UART_IT_IDLE

UART_IT_LBD

UART_IT_CTS

UART_IT_ERR

UART LIN Break Detection Length

UART_LINBREAKDETECTLENGTH_10B

UART_LINBREAKDETECTLENGTH_11B

UART Transfer Mode

UART_MODE_RX

UART_MODE_TX

UART_MODE_TX_RX

UART Over Sampling

UART_OVERSAMPLING_16

UART_OVERSAMPLING_8

UART Parity

UART_PARITY_NONE

UART_PARITY_EVEN

UART_PARITY_ODD

UART State

UART_STATE_DISABLE

UART_STATE_ENABLE

UART Number of Stop Bits

UART_STOPBITS_1

UART_STOPBITS_2

UART Wakeup Functions

UART_WAKEUPMETHOD_IDLELINE

UART_WAKEUPMETHOD_ADDRESSMARK

UART Word Length

UART_WORDLENGTH_8B

UART_WORDLENGTH_9B

46 HAL USART Generic Driver

46.1 USART Firmware driver registers structures

46.1.1 USART_InitTypeDef

Data Fields

- *uint32_t BaudRate*
- *uint32_t WordLength*
- *uint32_t StopBits*
- *uint32_t Parity*
- *uint32_t Mode*
- *uint32_t CLKPolarity*
- *uint32_t CLKPhase*
- *uint32_t CLKLastBit*

Field Documentation

- ***uint32_t USART_InitTypeDef::BaudRate***

This member configures the Usart communication baud rate. The baud rate is computed using the following formula:
IntegerDivider = ((PCLKx) / (8 * (husart->Init.BaudRate)))
FractionalDivider = ((IntegerDivider - ((uint32_t) IntegerDivider)) * 8)
+ 0.5

- ***uint32_t USART_InitTypeDef::WordLength***

Specifies the number of data bits transmitted or received in a frame. This parameter can be a value of **USART_Word_Length**

- ***uint32_t USART_InitTypeDef::StopBits***

Specifies the number of stop bits transmitted. This parameter can be a value of **USART_Stop_Bits**

- ***uint32_t USART_InitTypeDef::Parity***

Specifies the parity mode. This parameter can be a value of **USART_Parity**

Note: When parity is enabled, the computed parity is inserted at the MSB position of the transmitted data (9th bit when the word length is set to 9 data bits; 8th bit when the word length is set to 8 data bits).

- ***uint32_t USART_InitTypeDef::Mode***

Specifies whether the Receive or Transmit mode is enabled or disabled. This parameter can be a value of **USART_Mode**

- ***uint32_t USART_InitTypeDef::CLKPolarity***

Specifies the steady state of the serial clock. This parameter can be a value of **USART_Clock_Polarity**

- ***uint32_t USART_InitTypeDef::CLKPhase***

Specifies the clock transition on which the bit capture is made. This parameter can be a value of **USART_Clock_Phase**

- ***uint32_t USART_InitTypeDef::CLKLastBit***

Specifies whether the clock pulse corresponding to the last transmitted data bit (MSB) has to be output on the SCLK pin in synchronous mode. This parameter can be a value of **USART_Last_Bit**

46.1.2 USART_HandleTypeDef

Data Fields

- ***USART_TypeDef * Instance***

- ***USART_InitTypeDef Init***
- ***uint8_t * pTxBuffPtr***
- ***uint16_t TxXferSize***
- ***__IO uint16_t TxXferCount***
- ***uint8_t * pRxBuffPtr***
- ***uint16_t RxXferSize***
- ***__IO uint16_t RxXferCount***
- ***DMA_HandleTypeDef * hdmatx***
- ***DMA_HandleTypeDef * hdmarx***
- ***HAL_LockTypeDef Lock***
- ***__IO HAL_USART_StateTypeDef State***
- ***__IO uint32_t ErrorCode***

Field Documentation

- ***USART_TypeDef* USART_HandleTypeDef::Instance***
USART registers base address
- ***USART_InitTypeDef USART_HandleTypeDef::Init***
Usart communication parameters
- ***uint8_t* USART_HandleTypeDef::pTxBuffPtr***
Pointer to Usart Tx transfer Buffer
- ***uint16_t USART_HandleTypeDef::TxXferSize***
Usart Tx Transfer size
- ***__IO uint16_t USART_HandleTypeDef::TxXferCount***
Usart Tx Transfer Counter
- ***uint8_t* USART_HandleTypeDef::pRxBuffPtr***
Pointer to Usart Rx transfer Buffer
- ***uint16_t USART_HandleTypeDef::RxXferSize***
Usart Rx Transfer size
- ***__IO uint16_t USART_HandleTypeDef::RxXferCount***
Usart Rx Transfer Counter
- ***DMA_HandleTypeDef* USART_HandleTypeDef::hdmatx***
Usart Tx DMA Handle parameters
- ***DMA_HandleTypeDef* USART_HandleTypeDef::hdmarx***
Usart Rx DMA Handle parameters
- ***HAL_LockTypeDef USART_HandleTypeDef::Lock***
Locking object
- ***__IO HAL_USART_StateTypeDef USART_HandleTypeDef::State***
Usart communication state
- ***__IO uint32_t USART_HandleTypeDef::ErrorCode***
USART Error code

46.2 USART Firmware driver API description

46.2.1 How to use this driver

The USART HAL driver can be used as follows:

1. Declare a USART_HandleTypeDef handle structure.
2. Initialize the USART low level resources by implementing the HAL_USART_MspInit() API:
 - a. Enable the USARTx interface clock.
 - b. USART pins configuration:
 - Enable the clock for the USART GPIOs.
 - Configure the USART pins as alternate function pull-up.

- c. NVIC configuration if you need to use interrupt process (HAL_USART_Transmit_IT(), HAL_USART_Receive_IT() and HAL_USART_TransmitReceive_IT() APIs):
 - Configure the USARTx interrupt priority.
 - Enable the NVIC USART IRQ handle.
- d. DMA Configuration if you need to use DMA process (HAL_USART_Transmit_DMA() HAL_USART_Receive_DMA() and HAL_USART_TransmitReceive_DMA() APIs):
 - Declare a DMA handle structure for the Tx/Rx channel.
 - Enable the DMAx interface clock.
 - Configure the declared DMA handle structure with the required Tx/Rx parameters.
 - Configure the DMA Tx/Rx channel.
 - Associate the initialized DMA handle to the USART DMA Tx/Rx handle.
 - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx/Rx channel.
 - Configure the USARTx interrupt priority and enable the NVIC USART IRQ handle (used for last byte sending completion detection in DMA non circular mode)
- 3. Program the Baud Rate, Word Length, Stop Bit, Parity, Hardware flow control and Mode(Receiver/Transmitter) in the husart Init structure.
- 4. Initialize the USART registers by calling the HAL_USART_Init() API:
 - These APIs configures also the low level Hardware GPIO, CLOCK, CORTEX...etc) by calling the customized HAL_USART_MspInit(&husart) API. The specific USART interrupts (Transmission complete interrupt, RXNE interrupt and Error Interrupts) will be managed using the macros __HAL_USART_ENABLE_IT() and __HAL_USART_DISABLE_IT() inside the transmit and receive process.
- 5. Three operation modes are available within this driver :

Polling mode IO operation

- Send an amount of data in blocking mode using HAL_USART_Transmit()
- Receive an amount of data in blocking mode using HAL_USART_Receive()

Interrupt mode IO operation

- Send an amount of data in non blocking mode using HAL_USART_Transmit_IT()
- At transmission end of transfer HAL_USART_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_USART_TxCpltCallback
- Receive an amount of data in non blocking mode using HAL_USART_Receive_IT()
- At reception end of transfer HAL_USART_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_USART_RxCpltCallback
- In case of transfer Error, HAL_USART_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_USART_ErrorCallback

DMA mode IO operation

- Send an amount of data in non blocking mode (DMA) using HAL_USART_Transmit_DMA()
- At transmission end of half transfer HAL_USART_TxHalfCpltCallback is executed and user can add his own code by customization of function pointer HAL_USART_TxHalfCpltCallback

- At transmission end of transfer HAL_USART_TxCpltCallback is executed and user can add his own code by customization of function pointer HAL_USART_TxCpltCallback
- Receive an amount of data in non blocking mode (DMA) using HAL_USART_Receive_DMA()
- At reception end of half transfer HAL_USART_RxHalfCpltCallback is executed and user can add his own code by customization of function pointer HAL_USART_RxHalfCpltCallback
- At reception end of transfer HAL_USART_RxCpltCallback is executed and user can add his own code by customization of function pointer HAL_USART_RxCpltCallback
- In case of transfer Error, HAL_USART_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_USART_ErrorCallback
- Pause the DMA Transfer using HAL_USART_DMAPause()
- Resume the DMA Transfer using HAL_USART_DMAResume()
- Stop the DMA Transfer using HAL_USART_DMAStop()

USART HAL driver macros list

Below the list of most used macros in USART HAL driver.

- __HAL_USART_ENABLE: Enable the USART peripheral
- __HAL_USART_DISABLE: Disable the USART peripheral
- __HAL_USART_GET_FLAG : Check whether the specified USART flag is set or not
- __HAL_USART_CLEAR_FLAG : Clear the specified USART pending flag
- __HAL_USART_ENABLE_IT: Enable the specified USART interrupt
- __HAL_USART_DISABLE_IT: Disable the specified USART interrupt
- __HAL_USART_GET_IT_SOURCE: Check whether the specified USART interrupt has occurred or not



You can refer to the USART HAL driver header file for more useful macros

46.2.2 Initialization and Configuration functions

This subsection provides a set of functions allowing to initialize the USART in asynchronous and in synchronous modes.

- For the asynchronous mode only these parameters can be configured:
 - Baud Rate
 - Word Length
 - Stop Bit
 - Parity
 - USART polarity
 - USART phase
 - USART LastBit
 - Receiver/transmitter modes

The HAL_USART_Init() function follows the USART synchronous configuration procedure (details for the procedure are available in reference manual (RM0038)).

This section contains the following APIs:

- [**HAL_USART_Init\(\)**](#)
- [**HAL_USART_DeInit\(\)**](#)

- [*HAL_USART_MspInit\(\)*](#)
- [*HAL_USART_MspDeInit\(\)*](#)

46.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the USART synchronous data transfers.

The USART supports master mode only: it cannot receive or send data related to an input clock (SCLK is always an output).

1. There are two modes of transfer:
 - Blocking mode: The communication is performed in polling mode. The HAL status of all data processing is returned by the same function after finishing transfer.
 - No-Blocking mode: The communication is performed using Interrupts or DMA, These API's return the HAL status. The end of the data processing will be indicated through the dedicated USART IRQ when using Interrupt mode or the DMA IRQ when using DMA mode. The *HAL_USART_TxCpltCallback()*, *HAL_USART_RxCpltCallback()* and *HAL_USART_TxRxCpltCallback()* user callbacks will be executed respectively at the end of the transmit or Receive process. The *HAL_USART_ErrorCallback()* user callback will be executed when a communication error is detected
2. Blocking mode APIs are :
 - *HAL_USART_Transmit()* in simplex mode
 - *HAL_USART_Receive()* in full duplex receive only
 - *HAL_USART_TransmitReceive()* in full duplex mode
3. Non Blocking mode APIs with Interrupt are :
 - *HAL_USART_Transmit_IT()*in simplex mode
 - *HAL_USART_Receive_IT()* in full duplex receive only
 - *HAL_USART_TransmitReceive_IT()* in full duplex mode
 - *HAL_USART_IRQHandler()*
4. Non Blocking mode functions with DMA are :
 - *HAL_USART_Transmit_DMA()*in simplex mode
 - *HAL_USART_Receive_DMA()* in full duplex receive only
 - *HAL_USART_TransmitReceive_DMA()* in full duplex mode
 - *HAL_USART_DMAPause()*
 - *HAL_USART_DMAResume()*
 - *HAL_USART_DMAStop()*
5. A set of Transfer Complete Callbacks are provided in non Blocking mode:
 - *HAL_USART_TxHalfCpltCallback()*
 - *HAL_USART_TxCpltCallback()*
 - *HAL_USART_RxHalfCpltCallback()*
 - *HAL_USART_RxCpltCallback()*
 - *HAL_USART_ErrorCallback()*
 - *HAL_USART_TxRxCpltCallback()*

This section contains the following APIs:

- [*HAL_USART_Transmit\(\)*](#)
- [*HAL_USART_Receive\(\)*](#)
- [*HAL_USART_TransmitReceive\(\)*](#)
- [*HAL_USART_Transmit_IT\(\)*](#)
- [*HAL_USART_Receive_IT\(\)*](#)
- [*HAL_USART_TransmitReceive_IT\(\)*](#)
- [*HAL_USART_Transmit_DMA\(\)*](#)

- [`HAL_USART_Receive_DMA\(\)`](#)
- [`HAL_USART_TransmitReceive_DMA\(\)`](#)
- [`HAL_USART_DMAPause\(\)`](#)
- [`HAL_USART_DMAResume\(\)`](#)
- [`HAL_USART_DMAStop\(\)`](#)
- [`HAL_USART_IRQHandler\(\)`](#)
- [`HAL_USART_TxCpltCallback\(\)`](#)
- [`HAL_USART_TxHalfCpltCallback\(\)`](#)
- [`HAL_USART_RxCpltCallback\(\)`](#)
- [`HAL_USART_RxHalfCpltCallback\(\)`](#)
- [`HAL_USART_TxRxCpltCallback\(\)`](#)
- [`HAL_USART_ErrorCallback\(\)`](#)

46.2.4 Peripheral State and Errors functions

This subsection provides a set of functions allowing to return the State of USART communication process, return Peripheral Errors occurred during communication process

- `HAL_USART_GetState()` API can be helpful to check in run-time the state of the USART peripheral.
- `HAL_USART_GetError()` check in run-time errors that could be occurred during communication.

This section contains the following APIs:

- [`HAL_USART_GetState\(\)`](#)
- [`HAL_USART_GetError\(\)`](#)

46.2.5 Detailed description of functions

`HAL_USART_Init`

Function name	<code>HAL_StatusTypeDef HAL_USART_Init (USART_HandleTypeDef * huart)</code>
Function description	Initializes the USART mode according to the specified parameters in the USART_InitTypeDef and create the associated handle.
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	<ul style="list-style-type: none"> • HAL: status

`HAL_USART_DelInit`

Function name	<code>HAL_StatusTypeDef HAL_USART_DelInit (USART_HandleTypeDef * huart)</code>
Function description	Delinitializes the USART peripheral.
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_USART_MspInit

Function name	void HAL_USART_MspInit (USART_HandleTypeDef * husart)
Function description	USART MSP Init.
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	<ul style="list-style-type: none"> • None:

HAL_USART_MspDelInit

Function name	void HAL_USART_MspDelInit (USART_HandleTypeDef * husart)
Function description	USART MSP DelInit.
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	<ul style="list-style-type: none"> • None:

HAL_USART_Transmit

Function name	HAL_StatusTypeDef HAL_USART_Transmit (USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t Size, uint32_t Timeout)
Function description	Simplex Send an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module. • pTxData: Pointer to data buffer • Size: Amount of data to be sent • Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_USART_Receive

Function name	HAL_StatusTypeDef HAL_USART_Receive (USART_HandleTypeDef * husart, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)
Function description	Full-Duplex Receive an amount of data in blocking mode.
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module. • pRxData: Pointer to data buffer • Size: Amount of data to be received • Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_USART_TransmitReceive

Function name	HAL_StatusTypeDef HAL_USART_TransmitReceive (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)
Function description	Full-Duplex Send receive an amount of data in full-duplex mode (blocking mode).
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module. • pTxData: Pointer to data transmitted buffer • pRxData: Pointer to data received buffer • Size: Amount of data to be sent • Timeout: Timeout duration
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_USART_Transmit_IT

Function name	HAL_StatusTypeDef HAL_USART_Transmit_IT (USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t Size)
Function description	Simplex Send an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module. • pTxData: Pointer to data buffer • Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> • HAL: status
Notes	<ul style="list-style-type: none"> • The USART errors are not managed to avoid the overrun error.

HAL_USART_Receive_IT

Function name	HAL_StatusTypeDef HAL_USART_Receive_IT (USART_HandleTypeDef * husart, uint8_t * pRxData, uint16_t Size)
Function description	Simplex Receive an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> • husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module. • pRxData: Pointer to data buffer • Size: Amount of data to be received
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_USART_TransmitReceive_IT

Function name	HAL_StatusTypeDef HAL_USART_TransmitReceive_IT (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)
---------------	---

Function description	Full-Duplex Send receive an amount of data in full-duplex mode (non-blocking).
Parameters	<ul style="list-style-type: none"> husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module. pTxData: Pointer to data transmitted buffer pRxData: Pointer to data received buffer Size: Amount of data to be received
Return values	<ul style="list-style-type: none"> HAL: status

HAL_USART_Transmit_DMA

Function name	HAL_StatusTypeDef HAL_USART_Transmit_DMA (USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t Size)
Function description	Simplex Send an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module. pTxData: Pointer to data buffer Size: Amount of data to be sent
Return values	<ul style="list-style-type: none"> HAL: status

HAL_USART_Receive_DMA

Function name	HAL_StatusTypeDef HAL_USART_Receive_DMA (USART_HandleTypeDef * husart, uint8_t * pRxData, uint16_t Size)
Function description	Full-Duplex Receive an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module. pRxData: Pointer to data buffer Size: Amount of data to be received
Return values	<ul style="list-style-type: none"> HAL: status
Notes	<ul style="list-style-type: none"> The USART DMA transmit channel must be configured in order to generate the clock for the slave. When the USART parity is enabled (PCE = 1) the data received contain the parity bit.

HAL_USART_TransmitReceive_DMA

Function name	HAL_StatusTypeDef HAL_USART_TransmitReceive_DMA (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)
Function description	Full-Duplex Transmit Receive an amount of data in non-blocking mode.
Parameters	<ul style="list-style-type: none"> husart: Pointer to a USART_HandleTypeDef structure that

	contains the configuration information for the specified USART module.
• pTxData: Pointer to data transmitted buffer	
• pRxData: Pointer to data received buffer	
• Size: Amount of data to be received	
Return values	• HAL: status
Notes	• When the USART parity is enabled (PCE = 1) the data received contain the parity bit.

HAL_USART_DMAPause

Function name	HAL_StatusTypeDef HAL_USART_DMAPause (USART_HandleTypeDef * husart)
Function description	Pauses the DMA Transfer.
Parameters	• husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	• HAL: status

HAL_USART_DMAResume

Function name	HAL_StatusTypeDef HAL_USART_DMAResume (USART_HandleTypeDef * husart)
Function description	Resumes the DMA Transfer.
Parameters	• husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	• HAL: status

HAL_USART_DMAStop

Function name	HAL_StatusTypeDef HAL_USART_DMAStop (USART_HandleTypeDef * husart)
Function description	Stops the DMA Transfer.
Parameters	• husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	• HAL: status

HAL_USART_IRQHandler

Function name	void HAL_USART_IRQHandler (USART_HandleTypeDef * husart)
Function description	This function handles USART interrupt request.
Parameters	• husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.

Return values

- **None:**

HAL_USART_TxCpltCallback**Function name**

**void HAL_USART_TxCpltCallback (USART_HandleTypeDef *
husart)**

Function description

Tx Transfer completed callbacks.

Parameters

- **husart:** Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.

Return values

- **None:**

HAL_USART_TxHalfCpltCallback**Function name**

**void HAL_USART_TxHalfCpltCallback
(USART_HandleTypeDef * husart)**

Function description

Tx Half Transfer completed callbacks.

Parameters

- **husart:** Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.

Return values

- **None:**

HAL_USART_RxCpltCallback**Function name**

**void HAL_USART_RxCpltCallback (USART_HandleTypeDef *
husart)**

Function description

Rx Transfer completed callbacks.

Parameters

- **husart:** Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.

Return values

- **None:**

HAL_USART_RxHalfCpltCallback**Function name**

**void HAL_USART_RxHalfCpltCallback
(USART_HandleTypeDef * husart)**

Function description

Rx Half Transfer completed callbacks.

Parameters

- **husart:** Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.

Return values

- **None:**

HAL_USART_TxRxCpltCallback**Function name**

**void HAL_USART_TxRxCpltCallback (USART_HandleTypeDef
* husart)**

Function description

Tx/Rx Transfers completed callback for the non-blocking process.

Parameters	<ul style="list-style-type: none"> husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	<ul style="list-style-type: none"> None:

HAL_USART_ErrorCallback

Function name	void HAL_USART_ErrorCallback (USART_HandleTypeDef * husart)
Function description	USART error callbacks.
Parameters	<ul style="list-style-type: none"> husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	<ul style="list-style-type: none"> None:

HAL_USART_GetState

Function name	HAL_USART_StateTypeDef HAL_USART_GetState (USART_HandleTypeDef * husart)
Function description	Returns the USART state.
Parameters	<ul style="list-style-type: none"> husart: Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.
Return values	<ul style="list-style-type: none"> HAL: state

HAL_USART_GetError

Function name	uint32_t HAL_USART_GetError (USART_HandleTypeDef * husart)
Function description	Return the USART error code.
Parameters	<ul style="list-style-type: none"> husart: : pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART.
Return values	<ul style="list-style-type: none"> USART: Error Code

46.3 USART Firmware driver defines

46.3.1 USART

USART Clock

USART_CLOCK_DISABLE

USART_CLOCK_ENABLE

USART Clock Phase

USART_PHASE_1EDGE

USART_PHASE_2EDGE

USART Clock Polarity

`USART_POLARITY_LOW``USART_POLARITY_HIGH`***USART Error Codes***

<code>HAL_USART_ERROR_NONE</code>	No error
<code>HAL_USART_ERROR_PE</code>	Parity error
<code>HAL_USART_ERROR_NE</code>	Noise error
<code>HAL_USART_ERROR_FE</code>	frame error
<code>HAL_USART_ERROR_ORE</code>	Overrun error
<code>HAL_USART_ERROR_DMA</code>	DMA transfer error

USART Exported Macros`_HAL_USART_RESET_HANDLE_STATE`**Description:**

- Reset USART handle state.

Parameters:

- `_HANDLE_`: specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`_HAL_USART_GET_FLAG`**Description:**

- Check whether the specified USART flag is set or not.

Parameters:

- `_HANDLE_`: specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).
- `_FLAG_`: specifies the flag to check. This parameter can be one of the following values:
 - `USART_FLAG_TXE`: Transmit data register empty flag
 - `USART_FLAG_TC`: Transmission Complete flag
 - `USART_FLAG_RXNE`: Receive data register not empty flag
 - `USART_FLAG_IDLE`: Idle Line detection flag
 - `USART_FLAG_ORE`: OverRun Error flag
 - `USART_FLAG_NE`: Noise Error flag

- USART_FLAG_FE: Framing Error flag
- USART_FLAG_PE: Parity Error flag

Return value:

- The new state of __FLAG__ (TRUE or FALSE).

[__HAL_USART_CLEAR_FLAG](#)**Description:**

- Clear the specified USART pending flags.

Parameters:

- __HANDLE__: specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).
- __FLAG__: specifies the flag to check. This parameter can be any combination of the following values:
 - USART_FLAG_TC: Transmission Complete flag.
 - USART_FLAG_RXNE: Receive data register not empty flag.

Return value:

- None

Notes:

- PE (Parity error), FE (Framing error), NE (Noise error), ORE (OverRun error) and IDLE (Idle line detected) flags are cleared by software sequence: a read operation to USART_SR register followed by a read operation to USART_DR register. RXNE flag can be also cleared by a read to the USART_DR register. TC flag can be also cleared by software sequence: a read operation to USART_SR register followed by a write operation to USART_DR register. TXE flag is cleared only by a write to the USART_DR register.

[__HAL_USART_CLEAR_PEFLAG](#)**Description:**

- Clear the USART PE pending flag.

Parameters:

- __HANDLE__: specifies the USART Handle. USART Handle

selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

[__HAL_USART_CLEAR_FEFLAG](#)

Description:

- Clear the USART FE pending flag.

Parameters:

- [__HANDLE__](#): specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

[__HAL_USART_CLEAR_NEFLAG](#)

Description:

- Clear the USART NE pending flag.

Parameters:

- [__HANDLE__](#): specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

[__HAL_USART_CLEAR_OREFLAG](#)

Description:

- Clear the USART ORE pending flag.

Parameters:

- [__HANDLE__](#): specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

[__HAL_USART_CLEAR_IDLEFLAG](#)

Description:

- Clear the USART IDLE pending flag.

Parameters:

- [__HANDLE__](#): specifies the USART Handle. USART Handle

selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

_HAL_USART_ENABLE_IT

Description:

- Enable the specified Usart interrupts.

Parameters:

- HANDLE: specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).
- INTERRUPT: specifies the USART interrupt source to enable. This parameter can be one of the following values:
 - USART_IT_TXE: Transmit Data Register empty interrupt
 - USART_IT_TC: Transmission complete interrupt
 - USART_IT_RXNE: Receive Data register not empty interrupt
 - USART_IT_IDLE: Idle line detection interrupt
 - USART_IT_PE: Parity Error interrupt
 - USART_IT_ERR: Error interrupt(Frame error, noise error, overrun error)

Return value:

- None

_HAL_USART_DISABLE_IT

Description:

- Disable the specified Usart interrupts.

Parameters:

- HANDLE: specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).
- INTERRUPT: specifies the USART interrupt source to disable. This parameter can be one of the following values:
 - USART_IT_TXE: Transmit

- Data Register empty interrupt
- USART_IT_TC: Transmission complete interrupt
- USART_IT_RXNE: Receive Data register not empty interrupt
- USART_IT_IDLE: Idle line detection interrupt
- USART_IT_PE: Parity Error interrupt
- USART_IT_ERR: Error interrupt(Frame error, noise error, overrun error)

Return value:

- None

`__HAL_USART_GET_IT_SOURCE`**Description:**

- Check whether the specified Usart interrupt has occurred or not.

Parameters:

- `__HANDLE__`: specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).
- `__IT__`: specifies the USART interrupt source to check. This parameter can be one of the following values:
 - USART_IT_TXE: Transmit Data Register empty interrupt
 - USART_IT_TC: Transmission complete interrupt
 - USART_IT_RXNE: Receive Data register not empty interrupt
 - USART_IT_IDLE: Idle line detection interrupt
 - USART_IT_ERR: Error interrupt
 - USART_IT_PE: Parity Error interrupt

Return value:

- The: new state of `__IT__` (TRUE or FALSE).

`__HAL_USART_ONE_BIT_SAMPLE_ENABLE`**Description:**

- Enables the USART one bit sample method.

Parameters:

- `__HANDLE__`: specifies the USART Handle.

Return value:

- None

`__HAL_USART_ONE_BIT_SAMPLE_DISABLE`

- Disables the UART one bit sample method.

Parameters:

- `__HANDLE__`: specifies the USART Handle.

Return value:

- None

`__HAL_USART_ENABLE`

- Enable USART.

Parameters:

- `__HANDLE__`: specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

`__HAL_USART_DISABLE`

- Disable USART.

Parameters:

- `__HANDLE__`: specifies the USART Handle. USART Handle selects the USARTx peripheral (USART availability and x value depending on device).

Return value:

- None

USART Flags

`USART_FLAG_CTS`

`USART_FLAG_LBD`

`USART_FLAG_TXE`

`USART_FLAG_TC`

`USART_FLAG_RXNE`

`USART_FLAG_IDLE`

`USART_FLAG_ORE`

USART_FLAG_NE

USART_FLAG_FE

USART_FLAG_PE

USART Interrupts Definition

USART_IT_PE

USART_IT_TXE

USART_IT_TC

USART_IT_RXNE

USART_IT_IDLE

USART_IT_LBD

USART_IT_CTS

USART_IT_ERR

USART Last Bit

USART_LASTBIT_DISABLE

USART_LASTBIT_ENABLE

USART Mode

USART_MODE_RX

USART_MODE_TX

USART_MODE_TX_RX

USART NACK State

USART_NACK_ENABLE

USART_NACK_DISABLE

USART Parity

USART_PARITY_NONE

USART_PARITY_EVEN

USART_PARITY_ODD

USART Number of Stop Bits

USART_STOPBITS_1

USART_STOPBITS_0_5

USART_STOPBITS_2

USART_STOPBITS_1_5

USART Word Length

USART_WORDLENGTH_8B

USART_WORDLENGTH_9B

47 HAL WWDG Generic Driver

47.1 WWDG Firmware driver registers structures

47.1.1 WWDG_InitTypeDef

Data Fields

- *uint32_t Prescaler*
- *uint32_t Window*
- *uint32_t Counter*
- *uint32_t EWIMode*

Field Documentation

- ***uint32_t WWDG_InitTypeDef::Prescaler***
Specifies the prescaler value of the WWDG. This parameter can be a value of [**WWDG_Prescaler**](#)
- ***uint32_t WWDG_InitTypeDef::Window***
Specifies the WWDG window value to be compared to the downcounter. This parameter must be a number Min_Data = 0x40 and Max_Data = 0x7F
- ***uint32_t WWDG_InitTypeDef::Counter***
Specifies the WWDG free-running downcounter value. This parameter must be a number between Min_Data = 0x40 and Max_Data = 0x7F
- ***uint32_t WWDG_InitTypeDef::EWIMode***
Specifies if WWDG Early Wakeup Interupt is enable or not. This parameter can be a value of [**WWDG_EWI_Mode**](#)

47.1.2 WWDG_HandleTypeDef

Data Fields

- *WWDG_TypeDef * Instance*
- *WWDG_InitTypeDef Init*

Field Documentation

- ***WWDG_TypeDef* WWDG_HandleTypeDef::Instance***
Register base address
- ***WWDG_InitTypeDef WWDG_HandleTypeDef::Init***
WWDG required parameters

47.2 WWDG Firmware driver API description

47.2.1 WWDG specific features

Once enabled the WWDG generates a system reset on expiry of a programmed time period, unless the program refreshes the counter (T[6;0] downcounter) before reaching 0x3F value (i.e. a reset is generated when the counter value rolls over from 0x40 to 0x3F).

- An MCU reset is also generated if the counter value is refreshed before the counter has reached the refresh window value. This implies that the counter must be refreshed in a limited window.
- Once enabled the WWDG cannot be disabled except by a system reset.
- WWDRST flag in RCC_CSR register informs when a WWDG reset has occurred (check available with `__HAL_RCC_GET_FLAG(RCC_FLAG_WWDGRST)`).

- The WWDG downcounter input clock is derived from the APB clock divided by a programmable prescaler.
- WWDG downcounter clock (Hz) = PCLK1 / (4096 * Prescaler)
- WWDG timeout (ms) = (1000 * (T[5;0] + 1)) / (WWDG downcounter clock) where T[5;0] are the lowest 6 bits of downcounter.
- WWDG Counter refresh is allowed between the following limits :
 - min time (ms) = (1000 * (T[5;0] - Window)) / (WWDG downcounter clock)
 - max time (ms) = (1000 * (T[5;0] - 0x40)) / (WWDG downcounter clock)
- Min-max timeout value @80 MHz(PCLK1): ~51.2 us / ~26.22 ms
- The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device. In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions. Note:When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.
- Debug mode : When the microcontroller enters debug mode (core halted), the WWDG counter either continues to work normally or stops, depending on DBG_WWDG_STOP configuration bit in DBG module, accessible through __HAL_DBGMCU_FREEZE_WWDG() and __HAL_DBGMCU_UNFREEZE_WWDG() macros

47.2.2 How to use this driver

- Enable WWDG APB1 clock using __HAL_RCC_WWDG_CLK_ENABLE().
- Set the WWDG prescaler, refresh window, counter value and Early Wakeup Interrupt mode using using HAL_WWDG_Init() function. This enables WWDG peripheral and the downcounter starts downcounting from given counter value. Init function can be called again to modify all watchdog parameters, however if EWI mode has been set once, it can't be clear until next reset.
- The application program must refresh the WWDG counter at regular intervals during normal operation to prevent an MCU reset using HAL_WWDG_Refresh() function. This operation must occur only when the counter is lower than the window value already programmed.
- if Early Wakeup Interrupt mode is enable an interrupt is generated when the counter reaches 0x40. User can add his own code in weak function HAL_WWDG_EarlyWakeuCallback().

WWDG HAL driver macros list

Below the list of most used macros in WWDG HAL driver.

- __HAL_WWDG_GET_IT_SOURCE: Check the selected WWDG's interrupt source.
- __HAL_WWDG_GET_FLAG: Get the selected WWDG's flag status.
- __HAL_WWDG_CLEAR_FLAG: Clear the WWDG's pending flags.

47.2.3 Initialization and Configuration functions

This section provides functions allowing to:

- Initialize and start the WWDG according to the specified parameters in the WWDG_InitTypeDef of associated handle.
- Initialize the WWDG MSP.

This section contains the following APIs:

- [***HAL_WWDG_Init\(\)***](#)
- [***HAL_WWDG_MsInit\(\)***](#)

47.2.4 IO operation functions

This section provides functions allowing to:

- Refresh the WWDG.
- Handle WWDG interrupt request and associated function callback.

This section contains the following APIs:

- [***HAL_WWDG_Refresh\(\)***](#)
- [***HAL_WWDG_IRQHandler\(\)***](#)
- [***HAL_WWDG_EarlyWakeupCallback\(\)***](#)

47.2.5 Detailed description of functions

HAL_WWDG_Init

Function name	<code>HAL_StatusTypeDef HAL_WWDG_Init (WWDG_HandleTypeDef * hwdg)</code>
Function description	Initialize the WWDG according to the specified.
Parameters	<ul style="list-style-type: none"> • hwdg: pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_WWDG_MsInit

Function name	<code>void HAL_WWDG_MsInit (WWDG_HandleTypeDef * hwdg)</code>
Function description	Initialize the WWDG MSP.
Parameters	<ul style="list-style-type: none"> • hwdg: pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • When rewriting this function in user file, mechanism may be added to avoid multiple initialize when HAL_WWDG_Init function is called again to change parameters.

HAL_WWDG_Refresh

Function name	<code>HAL_StatusTypeDef HAL_WWDG_Refresh (WWDG_HandleTypeDef * hwdg)</code>
Function description	Refresh the WWDG.
Parameters	<ul style="list-style-type: none"> • hwdg: pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.
Return values	<ul style="list-style-type: none"> • HAL: status

HAL_WWDG_IRQHandler

Function name	void HAL_WWDG_IRQHandler (WWDG_HandleTypeDef * hwdg)
Function description	Handle WWDG interrupt request.
Parameters	<ul style="list-style-type: none"> hwdg: pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by calling HAL_WWDG_Init function with EWIMode set to WWDG_EWI_ENABLE. When the downcounter reaches the value 0x40, and EWI interrupt is generated and the corresponding Interrupt Service Routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

HAL_WWDG_EarlyWakeupCallback

Function name	void HAL_WWDG_EarlyWakeupCallback (WWDG_HandleTypeDef * hwdg)
Function description	WWDG Early Wakeup callback.
Parameters	<ul style="list-style-type: none"> hwdg: pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.
Return values	<ul style="list-style-type: none"> None:

47.3 WWDG Firmware driver defines

47.3.1 WWDG

WWDG Early Wakeup Interrupt Mode

WWDG_EWI_DISABLE EWI Disable

WWDG_EWI_ENABLE EWI Enable

WWDG Exported Macros

<code>_HAL_WWDG_ENABLE</code>	Description:
	<ul style="list-style-type: none"> Enable the WWDG peripheral.
	Parameters:
	<ul style="list-style-type: none"> <code>_HANDLE_</code>: WWDG handle
	Return value:
	<ul style="list-style-type: none"> None
<code>_HAL_WWDG_ENABLE_IT</code>	Description:
	<ul style="list-style-type: none"> Enable the WWDG early wakeup interrupt.

Parameters:

- `_HANDLE_`: WWdg handle
- `_INTERRUPT_`: specifies the interrupt to enable. This parameter can be one of the following values:
 - `WWDG_IT_EWI`: Early wakeup interrupt

Return value:

- None

Notes:

- Once enabled this interrupt cannot be disabled except by a system reset.

`__HAL_WWDG_GET_IT`**Description:**

- Check whether the selected WWdg interrupt has occurred or not.

Parameters:

- `_HANDLE_`: WWdg handle
- `_INTERRUPT_`: specifies the it to check. This parameter can be one of the following values:
 - `WWDG_FLAG_EWIF`: Early wakeup interrupt IT

Return value:

- The: new state of `WWDG_FLAG` (SET or RESET).

`__HAL_WWDG_CLEAR_IT`**Description:**

- Clear the WWdg interrupt pending bits.

Parameters:

- `_HANDLE_`: WWdg handle
- `_INTERRUPT_`: specifies the interrupt pending bit to clear. This parameter can be one of the following values:
 - `WWDG_FLAG_EWIF`: Early wakeup interrupt flag

`__HAL_WWDG_GET_FLAG`**Description:**

- Check whether the specified WWdg flag is set or not.

Parameters:

- `_HANDLE_`: WWdg handle
- `_FLAG_`: specifies the flag to check. This parameter can be one of the following values:
 - `WWDG_FLAG_EWIF`: Early wakeup interrupt flag

Return value:

- The: new state of `WWDG_FLAG` (SET or

RESET).

`_HAL_WWDG_CLEAR_FLAG`

Description:

- Clear the WWDG's pending flags.

Parameters:

- `_HANDLE_`: WWDG handle
- `_FLAG_`: specifies the flag to clear. This parameter can be one of the following values:
 - `WWDG_FLAG_EWIF`: Early wakeup interrupt flag

Return value:

- None

`_HAL_WWDG_GET_IT_SOURCE`

Description:

- Check whether the specified WWDG interrupt source is enabled or not.

Parameters:

- `_HANDLE_`: WWDG Handle.
- `_INTERRUPT_`: specifies the WWDG interrupt source to check. This parameter can be one of the following values:
 - `WWDG_IT_EWI`: Early Wakeup Interrupt

Return value:

- state: of `_INTERRUPT_` (TRUE or FALSE).

WWDG Flag definition

`WWDG_FLAG_EWIF` Early wakeup interrupt flag

WWDG Interrupt definition

`WWDG_IT_EWI` Early wakeup interrupt

WWDG Prescaler

`WWDG_PRESCALER_1` WWDG counter clock = (PCLK1/4096)/1

`WWDG_PRESCALER_2` WWDG counter clock = (PCLK1/4096)/2

`WWDG_PRESCALER_4` WWDG counter clock = (PCLK1/4096)/4

`WWDG_PRESCALER_8` WWDG counter clock = (PCLK1/4096)/8

48 LL ADC Generic Driver

48.1 ADC Firmware driver registers structures

48.1.1 LL_ADC_CommonInitTypeDef

Data Fields

- *uint32_t CommonClock*

Field Documentation

- *uint32_t LL_ADC_CommonInitTypeDef::CommonClock*

Set parameter common to several ADC: Clock source and prescaler. This parameter can be a value of [**ADC_LL_EC_COMMON_CLOCK_SOURCE**](#)

Note:On this STM32 serie, HSI RC oscillator is the only clock source for ADC. Therefore, HSI RC oscillator must be preliminarily enabled at RCC top level.On this STM32 serie, some clock ratio constraints between ADC clock and APB clock must be respected:In all cases: if APB clock frequency is too low compared ADC clock frequency, a delay between conversions must be inserted.If ADC group injected is used: ADC clock frequency should be lower than APB clock frequency /4 for resolution 12 or 10 bits, APB clock frequency /3 for resolution 8 bits, APB clock frequency /2 for resolution 6 bits. Refer to reference manual. This feature can be modified afterwards using unitary function **LL_ADC_SetCommonClock()**.

48.1.2 LL_ADC_InitTypeDef

Data Fields

- *uint32_t Resolution*
- *uint32_t DataAlignment*
- *uint32_t LowPowerMode*
- *uint32_t SequencersScanMode*

Field Documentation

- *uint32_t LL_ADC_InitTypeDef::Resolution*

Set ADC resolution. This parameter can be a value of [**ADC_LL_EC_RESOLUTION**](#)This feature can be modified afterwards using unitary function **LL_ADC_SetResolution()**.

- *uint32_t LL_ADC_InitTypeDef::DataAlignment*

Set ADC conversion data alignment. This parameter can be a value of [**ADC_LL_EC_DATA_ALIGN**](#)This feature can be modified afterwards using unitary function **LL_ADC_SetDataAlignment()**.

- *uint32_t LL_ADC_InitTypeDef::LowPowerMode*

Set ADC low power mode. This parameter can be a concatenation of a value of [**ADC_LL_EC_LP_MODE_AUTOWAIT**](#) and a value of [**ADC_LL_EC_LP_MODE_AUTOPOWEROFF**](#)This feature can be modified afterwards using unitary function **LL_ADC_SetLowPowerModeAutoWait()** and **LL_ADC_SetLowPowerModeAutoPowerOff()**.

- *uint32_t LL_ADC_InitTypeDef::SequencersScanMode*

Set ADC scan selection. This parameter can be a value of [**ADC_LL_EC_SCAN_SELECTION**](#)This feature can be modified afterwards using unitary function **LL_ADC_SetSequencersScanMode()**.

48.1.3 LL_ADC_REG_InitTypeDef

Data Fields

- *uint32_t TriggerSource*
- *uint32_t SequencerLength*
- *uint32_t SequencerDiscont*
- *uint32_t ContinuousMode*
- *uint32_t DMATransfer*

Field Documentation

- *uint32_t LL_ADC_REG_InitTypeDef::TriggerSource*

Set ADC group regular conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line). This parameter can be a value of

[**ADC_LL_EC_REG_TRIGGER_SOURCE**](#)

Note:On this STM32 serie, setting of external trigger edge is performed using function **LL_ADC_REG_StartConversionExtTrig()**. This feature can be modified afterwards using unitary function **LL_ADC_REG_SetTriggerSource()**.

- *uint32_t LL_ADC_REG_InitTypeDef::SequencerLength*

Set ADC group regular sequencer length. This parameter can be a value of

[**ADC_LL_EC_REG_SEQ_SCAN_LENGTH**](#)

Note:This parameter is discarded if scan mode is disabled (refer to parameter 'ADC_SequencersScanMode'). This feature can be modified afterwards using unitary function **LL_ADC_REG_SetSequencerLength()**.

- *uint32_t LL_ADC_REG_InitTypeDef::SequencerDiscont*

Set ADC group regular sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks. This parameter can be a value of [**ADC_LL_EC_REG_SEQ_DISCONT_MODE**](#)

Note:This parameter has an effect only if group regular sequencer is enabled (scan length of 2 ranks or more). This feature can be modified afterwards using unitary function **LL_ADC_REG_SetSequencerDiscont()**.

- *uint32_t LL_ADC_REG_InitTypeDef::ContinuousMode*

Set ADC continuous conversion mode on ADC group regular, whether ADC conversions are performed in single mode (one conversion per trigger) or in continuous mode (after the first trigger, following conversions launched successively automatically). This parameter can be a value of

[**ADC_LL_EC_REG_CONTINUOUS_MODE**](#) Note: It is not possible to enable both

ADC group regular continuous mode and discontinuous mode. This feature can be modified afterwards using unitary function **LL_ADC_REG_SetContinuousMode()**.

- *uint32_t LL_ADC_REG_InitTypeDef::DMATransfer*

Set ADC group regular conversion data transfer: no transfer or transfer by DMA, and DMA requests mode. This parameter can be a value of

[**ADC_LL_EC_REG_DMA_TRANSFER**](#) This feature can be modified afterwards using unitary function **LL_ADC_REG_SetDMATransfer()**.

48.1.4 LL_ADC_INJ_InitTypeDef

Data Fields

- *uint32_t TriggerSource*
- *uint32_t SequencerLength*
- *uint32_t SequencerDiscont*
- *uint32_t TrigAuto*

Field Documentation

- ***uint32_t LL_ADC_INJ_InitTypeDef::TriggerSource***
Set ADC group injected conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line). This parameter can be a value of ***ADC_LL_EC_INJ_TRIGGER_SOURCE***
Note:On this STM32 serie, setting of external trigger edge is performed using function ***LL_ADC_INJ_StartConversionExtTrig()***. This feature can be modified afterwards using unitary function ***LL_ADC_INJ_SetTriggerSource()***.
- ***uint32_t LL_ADC_INJ_InitTypeDef::SequencerLength***
Set ADC group injected sequencer length. This parameter can be a value of ***ADC_LL_EC_INJ_SEQ_SCAN_LENGTH***
Note:This parameter is discarded if scan mode is disabled (refer to parameter 'ADC_SequencersScanMode'). This feature can be modified afterwards using unitary function ***LL_ADC_INJ_SetSequencerLength()***.
- ***uint32_t LL_ADC_INJ_InitTypeDef::SequencerDiscont***
Set ADC group injected sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks. This parameter can be a value of ***ADC_LL_EC_INJ_SEQ_DISCONT_MODE***
Note:This parameter has an effect only if group injected sequencer is enabled (scan length of 2 ranks or more). This feature can be modified afterwards using unitary function ***LL_ADC_INJ_SetSequencerDiscont()***.
- ***uint32_t LL_ADC_INJ_InitTypeDef::TrigAuto***
Set ADC group injected conversion trigger: independent or from ADC group regular. This parameter can be a value of ***ADC_LL_EC_INJ_TRIG_AUTO*** Note: This parameter must be set to set to independent trigger if injected trigger source is set to an external trigger.This feature can be modified afterwards using unitary function ***LL_ADC_INJ_SetTrigAuto()***.

48.2 ADC Firmware driver API description

48.2.1 Detailed description of functions

LL_ADC_DMA_GetRegAddr

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_DMA_GetRegAddr (ADC_TypeDef * ADCx, uint32_t Register)</code>
Function description	Function to help to configure DMA transfer from ADC: retrieve the ADC register address from ADC instance and a list of ADC registers intended to be used (most commonly) with DMA transfer.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Register: This parameter can be one of the following values: <ul style="list-style-type: none"> – <code>LL_ADC_REG_REGULAR_DATA</code>
Return values	<ul style="list-style-type: none"> • ADC: register address
Notes	<ul style="list-style-type: none"> • These ADC registers are data registers: when ADC conversion data is available in ADC data registers, ADC generates a DMA transfer request. • This macro is intended to be used with LL DMA driver, refer to function "LL_DMA_ConfigAddresses()". Example: <code>LL_DMA_ConfigAddresses(DMA1, LL_DMA_CHANNEL_1, LL_ADC_DMA_GetRegAddr(ADC1, LL_ADC_REG_REGULAR_DATA), (uint32_t)&< array or variable >, LL_DMA_DIRECTION_PERIPH_TO_MEMORY);</code> • For devices with several ADC: in multimode, some devices

Reference Manual to LL API cross reference:	use a different data register outside of ADC instance scope (common data register). This macro manages this register difference, only ADC instance has to be set as parameter.
	<ul style="list-style-type: none"> • DR DATA LL_ADC_DMA_GetRegAddr

LL_ADC_SetCommonClock

Function name	<code>__STATIC_INLINE void LL_ADC_SetCommonClock(ADC_Common_TypeDef * ADCxy_COMMON, uint32_t CommonClock)</code>
Function description	Set parameter common to several ADC: Clock source and prescaler.
Parameters	<ul style="list-style-type: none"> • ADCxy_COMMON: ADC common instance (can be set directly from CMSIS definition or by using helper macro <code>__LL_ADC_COMMON_INSTANCE()</code>) • CommonClock: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_ADC_CLOCK_ASYNC_DIV1</code> - <code>LL_ADC_CLOCK_ASYNC_DIV2</code> - <code>LL_ADC_CLOCK_ASYNC_DIV4</code>
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • On this STM32 serie, HSI RC oscillator is the only clock source for ADC. Therefore, HSI RC oscillator must be preliminarily enabled at RCC top level. • On this STM32 serie, some clock ratio constraints between ADC clock and APB clock must be respected: In all cases: if APB clock frequency is too low compared ADC clock frequency, a delay between conversions must be inserted. If ADC group injected is used: ADC clock frequency should be lower than APB clock frequency /4 for resolution 12 or 10 bits, APB clock frequency /3 for resolution 8 bits, APB clock frequency /2 for resolution 6 bits. Refer to reference manual.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR ADCPRE LL_ADC_SetCommonClock

LL_ADC_GetCommonClock

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_GetCommonClock(ADC_Common_TypeDef * ADCxy_COMMON)</code>
Function description	Get parameter common to several ADC: Clock source and prescaler.
Parameters	<ul style="list-style-type: none"> • ADCxy_COMMON: ADC common instance (can be set directly from CMSIS definition or by using helper macro <code>__LL_ADC_COMMON_INSTANCE()</code>)
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - <code>LL_ADC_CLOCK_ASYNC_DIV1</code> - <code>LL_ADC_CLOCK_ASYNC_DIV2</code>

- LL_ADC_CLOCK_ASYNC_DIV4
 - CCR ADCPRE LL_ADC_GetCommonClock
- Reference Manual to
LL API cross
reference:

LL_ADC_SetCommonPathInternalCh

Function name	<code>__STATIC_INLINE void LL_ADC_SetCommonPathInternalCh(ADC_Common_TypeDef * ADCxy_COMMON, uint32_t PathInternal)</code>
Function description	Set parameter common to several ADC: measurement path to internal channels (VrefInt, temperature sensor, ...).
Parameters	<ul style="list-style-type: none"> • ADCxy_COMMON: ADC common instance (can be set directly from CMSIS definition or by using helper macro <code>__LL_ADC_COMMON_INSTANCE()</code>) • PathInternal: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_ADC_PATH_INTERNAL_NONE - LL_ADC_PATH_INTERNAL_VREFINT - LL_ADC_PATH_INTERNAL_TEMPSENSOR
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • One or several values can be selected. Example: (LL_ADC_PATH_INTERNAL_VREFINT LL_ADC_PATH_INTERNAL_TEMPSENSOR) • Stabilization time of measurement path to internal channel: After enabling internal paths, before starting ADC conversion, a delay is required for internal voltage reference and temperature sensor stabilization time. Refer to device datasheet. Refer to literal LL_ADC_DELAY_VREFINT_STAB_US. Refer to literal LL_ADC_DELAY_TEMPSENSOR_STAB_US. • ADC internal channel sampling time constraint: For ADC conversion of internal channels, a sampling time minimum value is required. Refer to device datasheet.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR TSVREFE LL_ADC_SetCommonPathInternalCh

LL_ADC_GetCommonPathInternalCh

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_GetCommonPathInternalCh(ADC_Common_TypeDef * ADCxy_COMMON)</code>
Function description	Get parameter common to several ADC: measurement path to internal channels (VrefInt, temperature sensor, ...).
Parameters	<ul style="list-style-type: none"> • ADCxy_COMMON: ADC common instance (can be set directly from CMSIS definition or by using helper macro <code>__LL_ADC_COMMON_INSTANCE()</code>)
Return values	<ul style="list-style-type: none"> • Returned: value can be a combination of the following values:

	<ul style="list-style-type: none"> - LL_ADC_PATH_INTERNAL_NONE - LL_ADC_PATH_INTERNAL_VREFINT - LL_ADC_PATH_INTERNAL_TEMPSENSOR
Notes	<ul style="list-style-type: none"> • One or several values can be selected. Example: (LL_ADC_PATH_INTERNAL_VREFINT LL_ADC_PATH_INTERNAL_TEMPSENSOR)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR TSVREFE LL_ADC_GetCommonPathInternalCh

LL_ADC_SetResolution

Function name	<code>_STATIC_INLINE void LL_ADC_SetResolution(ADC_TypeDef * ADCx, uint32_t Resolution)</code>
Function description	Set ADC resolution.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Resolution: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_RESOLUTION_12B - LL_ADC_RESOLUTION_10B - LL_ADC_RESOLUTION_8B - LL_ADC_RESOLUTION_6B
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 RES LL_ADC_SetResolution

LL_ADC_GetResolution

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_GetResolution(ADC_TypeDef * ADCx)</code>
Function description	Get ADC resolution.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_RESOLUTION_12B - LL_ADC_RESOLUTION_10B - LL_ADC_RESOLUTION_8B - LL_ADC_RESOLUTION_6B
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 RES LL_ADC_GetResolution

LL_ADC_SetDataAlignment

Function name	<code>_STATIC_INLINE void LL_ADC_SetDataAlignment(ADC_TypeDef * ADCx, uint32_t DataAlignment)</code>
Function description	Set ADC conversion data alignment.

Parameters	<ul style="list-style-type: none"> ADCx: ADC instance DataAlignment: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_DATA_ALIGN_RIGHT – LL_ADC_DATA_ALIGN_LEFT
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Refer to reference manual for alignments formats dependencies to ADC resolutions.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ALIGN LL_ADC_SetDataAlignment

LL_ADC_GetDataAlignment

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_GetDataAlignment(ADC_TypeDef * ADCx)</code>
Function description	Get ADC conversion data alignment.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_DATA_ALIGN_RIGHT – LL_ADC_DATA_ALIGN_LEFT
Notes	<ul style="list-style-type: none"> Refer to reference manual for alignments formats dependencies to ADC resolutions.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ALIGN LL_ADC_SetDataAlignment

LL_ADC_SetLowPowerModeAutoWait

Function name	<code>_STATIC_INLINE void LL_ADC_SetLowPowerModeAutoWait(ADC_TypeDef * ADCx, uint32_t LowPowerModeAutoWait)</code>
Function description	Set ADC low power mode auto wait.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance LowPowerModeAutoWait: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_LP_AUTOWAIT_NONE – LL_ADC_LP_AUTOWAIT – LL_ADC_LP_AUTOWAIT_7_APBCLOCKCYCLES – LL_ADC_LP_AUTOWAIT_15_APBCLOCKCYCLES – LL_ADC_LP_AUTOWAIT_31_APBCLOCKCYCLES – LL_ADC_LP_AUTOWAIT_63_APBCLOCKCYCLES – LL_ADC_LP_AUTOWAIT_127_APBCLOCKCYCLES – LL_ADC_LP_AUTOWAIT_255_APBCLOCKCYCLES
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Description of ADC low power modes: ADC low power mode "auto wait": Dynamic low power mode, ADC conversions occurrences are limited to the minimum necessary in order to reduce power consumption. New ADC conversion starts only

when the previous unitary conversion data (for ADC group regular) or previous sequence conversions data (for ADC group injected) has been retrieved by user software. In the meantime, ADC remains idle: does not performs any other conversion. This mode allows to automatically adapt the ADC conversions triggers to the speed of the software that reads the data. Moreover, this avoids risk of overrun for low frequency applications. How to use this low power mode: Do not use with interruption or DMA since these modes have to clear immediately the EOC flag to free the IRQ vector sequencer. Do use with polling: 1. Start conversion, 2. Later on, when conversion data is needed: poll for end of conversion to ensure that conversion is completed and retrieve ADC conversion data. This will trig another ADC conversion start. ADC low power mode "auto power-off": refer to function LL_ADC_SetLowPowerModeAutoPowerOff().

- With ADC low power mode "auto wait", the ADC conversion data read is corresponding to previous ADC conversion start, independently of delay during which ADC was idle. Therefore, the ADC conversion data may be outdated: does not correspond to the current voltage level on the selected ADC channel.
- CR2 DELS LL_ADC_SetLowPowerModeAutoWait

Reference Manual
to LL API cross
reference:

LL_ADC_GetLowPowerModeAutoWait

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_GetLowPowerModeAutoWait (ADC_TypeDef * ADCx)</code>
Function description	Get ADC low power mode auto wait.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_LP_AUTOWAIT_NONE - LL_ADC_LP_AUTOWAIT - LL_ADC_LP_AUTOWAIT_7_APBCLOCKCYCLES - LL_ADC_LP_AUTOWAIT_15_APBCLOCKCYCLES - LL_ADC_LP_AUTOWAIT_31_APBCLOCKCYCLES - LL_ADC_LP_AUTOWAIT_63_APBCLOCKCYCLES - LL_ADC_LP_AUTOWAIT_127_APBCLOCKCYCLES - LL_ADC_LP_AUTOWAIT_255_APBCLOCKCYCLES
Notes	<ul style="list-style-type: none"> Description of ADC low power modes: ADC low power mode "auto wait": Dynamic low power mode, ADC conversions occurrences are limited to the minimum necessary in order to reduce power consumption. New ADC conversion starts only when the previous unitary conversion data (for ADC group regular) or previous sequence conversions data (for ADC group injected) has been retrieved by user software. In the meantime, ADC remains idle: does not performs any other conversion. This mode allows to automatically adapt the ADC conversions triggers to the speed of the software that reads the data. Moreover, this avoids risk of overrun for low

frequency applications. How to use this low power mode: Do not use with interruption or DMA since these modes have to clear immediately the EOC flag to free the IRQ vector sequencer. Do use with polling: 1. Start conversion, 2. Later on, when conversion data is needed: poll for end of conversion to ensure that conversion is completed and retrieve ADC conversion data. This will trig another ADC conversion start. ADC low power mode "auto power-off": refer to function LL_ADC_SetLowPowerModeAutoPowerOff().

- With ADC low power mode "auto wait", the ADC conversion data read is corresponding to previous ADC conversion start, independently of delay during which ADC was idle. Therefore, the ADC conversion data may be outdated: does not correspond to the current voltage level on the selected ADC channel.
- CR2 DELS LL_ADC_GetLowPowerModeAutoWait

Reference Manual
to LL API cross
reference:

LL_ADC_SetLowPowerModeAutoPowerOff

Function name	<code>_STATIC_INLINE void LL_ADC_SetLowPowerModeAutoPowerOff(ADC_TypeDef * ADCx, uint32_t LowPowerModeAutoPowerOff)</code>
Function description	Set ADC low power mode auto power-off.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance LowPowerModeAutoPowerOff: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_LP_AUTOPOWEROFF_NONE - LL_ADC_LP_AUTOPOWEROFF_IDLE_PHASE - LL_ADC_LP_AUTOPOWEROFF_AUTOWAIT_PHASE - LL_ADC_LP_AUTOPOWEROFF_IDLE_AUTOWAIT_PHASES
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Description of ADC low power modes: ADC low power mode "auto wait": refer to function LL_ADC_SetLowPowerModeAutoWait(). ADC low power mode "auto power-off": the ADC automatically powers-off after a conversion and automatically wakes up when a new conversion is triggered (with startup time between trigger and start of sampling). This feature can be combined with low power mode "auto wait".
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 PDI LL_ADC_GetLowPowerModeAutoPowerOff CR1 PDD LL_ADC_GetLowPowerModeAutoPowerOff

LL_ADC_GetLowPowerModeAutoPowerOff

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_GetLowPowerModeAutoPowerOff(ADC_TypeDef * ADCx)</code>
Function	Get ADC low power mode auto power-off.

description	
Parameter s	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_LP_AUTOPOWEROFF_NONE - LL_ADC_LP_AUTOPOWEROFF_IDLE_PHASE - LL_ADC_LP_AUTOPOWEROFF_AUTOWAIT_PHASE - LL_ADC_LP_AUTOPOWEROFF_IDLE_AUTOWAIT_PHASES
Notes	<ul style="list-style-type: none"> Description of ADC low power modes: ADC low power mode "auto wait": refer to function LL_ADC_SetLowPowerModeAutoWait().ADC low power mode "auto power-off": the ADC automatically powers-off after a conversion and automatically wakes up when a new conversion is triggered (with startup time between trigger and start of sampling). This feature can be combined with low power mode "auto wait".
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 PDI LL_ADC_GetLowPowerModeAutoPowerOff CR1 PDD LL_ADC_GetLowPowerModeAutoPowerOff

LL_ADC_SetSequencersScanMode

Function name	<code>__STATIC_INLINE void LL_ADC_SetSequencersScanMode(ADC_TypeDef * ADCx, uint32_t ScanMode)</code>
Function description	Set ADC sequencers scan mode, for all ADC groups (group regular, group injected).
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance ScanMode: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_SEQ_SCAN_DISABLE - LL_ADC_SEQ_SCAN_ENABLE
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> According to sequencers scan mode : If disabled: ADC conversion is performed in unitary conversion mode (one channel converted, that defined in rank 1). Configuration of sequencers of all ADC groups (sequencer scan length, ...) is discarded: equivalent to scan length of 1 rank.If enabled: ADC conversions are performed in sequence conversions mode, according to configuration of sequencers of each ADC group (sequencer scan length, ...). Refer to function LL_ADC_REG_SetSequencerLength() and to function LL_ADC_INJ_SetSequencerLength(). On this STM32 serie, setting of this feature is conditioned to ADC state: ADC must be disabled or enabled without conversion on going on either groups regular or injected.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 SCAN LL_ADC_SetSequencersScanMode

LL_ADC_GetSequencersScanMode

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_GetSequencersScanMode (ADC_TypeDef * ADCx)</code>
Function description	Get ADC sequencers scan mode, for all ADC groups (group regular, group injected).
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_SEQ_SCAN_DISABLE – LL_ADC_SEQ_SCAN_ENABLE
Notes	<ul style="list-style-type: none"> • According to sequencers scan mode : If disabled: ADC conversion is performed in unitary conversion mode (one channel converted, that defined in rank 1). Configuration of sequencers of all ADC groups (sequencer scan length, ...) is discarded: equivalent to scan length of 1 rank. If enabled: ADC conversions are performed in sequence conversions mode, according to configuration of sequencers of each ADC group (sequencer scan length, ...). Refer to function <code>LL_ADC_REG_SetSequencerLength()</code> and to function <code>LL_ADC_INJ_SetSequencerLength()</code>.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 SCAN LL_ADC_GetSequencersScanMode

LL_ADC_SetChannelsBank

Function name	<code>__STATIC_INLINE void LL_ADC_SetChannelsBank (ADC_TypeDef * ADCx, uint32_t ChannelsBank)</code>
Function description	Set ADC channels bank.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • ChannelsBank: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_CHANNELS_BANK_A – LL_ADC_CHANNELS_BANK_B
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bank selected applies to ADC scope, on all channels (independently of channel mapped on ADC group regular or group injected). • Banks availability depends on devices categories. • On this STM32 serie, setting of this feature is conditioned to ADC state: ADC must be disabled or enabled without conversion on going on either groups regular or injected.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ADC_CFG LL_ADC_SetChannelsBank

LL_ADC_GetChannelsBank

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_GetChannelsBank</code>
---------------	--

(ADC_TypeDef * ADCx)

Function description	Get ADC channels bank.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_CHANNELS_BANK_A - LL_ADC_CHANNELS_BANK_B
Notes	<ul style="list-style-type: none"> • Bank selected applies to ADC scope, on all channels (independently of channel mapped on ADC group regular or group injected). • Banks availability depends on devices categories.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ADC_CFG LL_ADC_GetChannelsBank

LL_ADC_REG_SetTriggerSource

Function name	_STATIC_INLINE void LL_ADC_REG_SetTriggerSource(ADC_TypeDef * ADCx, uint32_t TriggerSource)
Function description	Set ADC group regular conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line).
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • TriggerSource: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_REG_TRIG_SOFTWARE - LL_ADC_REG_TRIG_EXT_TIM2_TRGO - LL_ADC_REG_TRIG_EXT_TIM2_CH3 - LL_ADC_REG_TRIG_EXT_TIM3_TRGO - LL_ADC_REG_TRIG_EXT_TIM2_CH2 - LL_ADC_REG_TRIG_EXT_TIM3_CH1 - LL_ADC_REG_TRIG_EXT_TIM3_CH3 - LL_ADC_REG_TRIG_EXT_TIM4_TRGO - LL_ADC_REG_TRIG_EXT_TIM4_CH4 - LL_ADC_REG_TRIG_EXT_TIM6_TRGO - LL_ADC_REG_TRIG_EXT_TIM9_CH2 - LL_ADC_REG_TRIG_EXT_TIM9_TRGO - LL_ADC_REG_TRIG_EXT EXTI_LINE11
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • On this STM32 serie, setting of external trigger edge is performed using function LL_ADC_REG_StartConversionExtTrig(). • Availability of parameters of trigger sources from timer depends on timers availability on the selected device.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 EXTSEL LL_ADC_REG_SetTriggerSource • CR2 EXTEEN LL_ADC_REG_SetTriggerSource

LL_ADC_REG_GetTriggerSource

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_REG_GetTriggerSource(ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line).
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_REG_TRIG_SOFTWARE - LL_ADC_REG_TRIG_EXT_TIM2_TRGO - LL_ADC_REG_TRIG_EXT_TIM2_CH3 - LL_ADC_REG_TRIG_EXT_TIM3_TRGO - LL_ADC_REG_TRIG_EXT_TIM2_CH2 - LL_ADC_REG_TRIG_EXT_TIM3_CH1 - LL_ADC_REG_TRIG_EXT_TIM3_CH3 - LL_ADC_REG_TRIG_EXT_TIM4_TRGO - LL_ADC_REG_TRIG_EXT_TIM4_CH4 - LL_ADC_REG_TRIG_EXT_TIM6_TRGO - LL_ADC_REG_TRIG_EXT_TIM9_CH2 - LL_ADC_REG_TRIG_EXT_TIM9_TRGO - LL_ADC_REG_TRIG_EXT_EXTI_LINE11
Notes	<ul style="list-style-type: none"> • To determine whether group regular trigger source is internal (SW start) or external, without detail of which peripheral is selected as external trigger, (equivalent to "if(LL_ADC_REG_GetTriggerSource(ADC1) == LL_ADC_REG_TRIG_SOFTWARE)" use function LL_ADC_REG_IsTriggerSourceSWStart. • Availability of parameters of trigger sources from timer depends on timers availability on the selected device.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 EXTSEL LL_ADC_REG_GetTriggerSource • CR2 EXTEN LL_ADC_REG_GetTriggerSource

LL_ADC_REG_IsTriggerSourceSWStart

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_REG_IsTriggerSourceSWStart (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion trigger source internal (SW start) or external.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Value: "0" if trigger source external trigger Value "1" if trigger source SW start.
Notes	<ul style="list-style-type: none"> • In case of group regular trigger source set to external trigger, to determine which peripheral is selected as external trigger, use function LL_ADC_REG_GetTriggerSource().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 EXTEN LL_ADC_REG_IsTriggerSourceSWStart

LL_ADC_REG_GetTriggerEdge

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_REG_GetTriggerEdge(ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion trigger polarity.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_REG_TRIG_EXT_RISING - LL_ADC_REG_TRIG_EXT_FALLING - LL_ADC_REG_TRIG_EXT_RISINGFALLING
Notes	<ul style="list-style-type: none"> • Applicable only for trigger source set to external trigger. • On this STM32 serie, setting of external trigger edge is performed using function <code>LL_ADC_REG_StartConversionExtTrig()</code>.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 EXTEN <code>LL_ADC_REG_GetTriggerEdge</code>

LL_ADC_REG_SetSequencerLength

Function name	<code>__STATIC_INLINE void LL_ADC_REG_SetSequencerLength(ADC_TypeDef * ADCx, uint32_t SequencerNbRanks)</code>
Function description	Set ADC group regular sequencer length and scan direction.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • SequencerNbRanks: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_REG_SEQ_SCAN_DISABLE - LL_ADC_REG_SEQ_SCAN_ENABLE_2RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_3RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_4RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_5RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_6RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_7RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_8RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_9RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_10RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_11RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_12RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_13RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_14RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_15RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_16RANKS
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Description of ADC group regular sequencer features: For devices with sequencer fully configurable (function "LL_ADC_REG_SetSequencerRanks()" available): sequencer length and each rank affectation to a channel are configurable. This function performs configuration of: Sequence length: Number of ranks in the scan

sequence.Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from rank 1 to rank n). Sequencer ranks are selected using function "LL_ADC_REG_SetSequencerRanks()". For devices with sequencer not fully configurable (function "LL_ADC_REG_SetSequencerChannels()" available): sequencer length and each rank affection to a channel are defined by channel number. This function performs configuration of: Sequence length: Number of ranks in the scan sequence is defined by number of channels set in the sequence, rank of each channel is fixed by channel HW number. (channel 0 fixed on rank 0, channel 1 fixed on rank1, ...).Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from lowest channel number to highest channel number). Sequencer ranks are selected using function "LL_ADC_REG_SetSequencerChannels()".

- On this STM32 serie, group regular sequencer configuration is conditioned to ADC instance sequencer mode. If ADC instance sequencer mode is disabled, sequencers of all groups (group regular, group injected) can be configured but their execution is disabled (limited to rank 1). Refer to function LL_ADC_SetSequencersScanMode().
- Sequencer disabled is equivalent to sequencer of 1 rank: ADC conversion on only 1 channel.
- SQR1 L LL_ADC_REG_SetSequencerLength

Reference Manual to
LL API cross
reference:

LL_ADC_REG_GetSequencerLength

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_REG_GetSequencerLength (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular sequencer length and scan direction.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_REG_SEQ_SCAN_DISABLE - LL_ADC_REG_SEQ_SCAN_ENABLE_2RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_3RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_4RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_5RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_6RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_7RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_8RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_9RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_10RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_11RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_12RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_13RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_14RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_15RANKS - LL_ADC_REG_SEQ_SCAN_ENABLE_16RANKS

Notes

- Description of ADC group regular sequencer features: For devices with sequencer fully configurable (function "LL_ADC_REG_SetSequencerRanks()" available): sequencer length and each rank affectation to a channel are configurable. This function retrieves: Sequence length: Number of ranks in the scan sequence. Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from rank 1 to rank n). Sequencer ranks are selected using function "LL_ADC_REG_SetSequencerRanks()". For devices with sequencer not fully configurable (function "LL_ADC_REG_SetSequencerChannels()" available): sequencer length and each rank affectation to a channel are defined by channel number. This function retrieves: Sequence length: Number of ranks in the scan sequence is defined by number of channels set in the sequence, rank of each channel is fixed by channel HW number. (channel 0 fixed on rank 0, channel 1 fixed on rank1, ...). Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from lowest channel number to highest channel number). Sequencer ranks are selected using function "LL_ADC_REG_SetSequencerChannels()".
- On this STM32 serie, group regular sequencer configuration is conditioned to ADC instance sequencer mode. If ADC instance sequencer mode is disabled, sequencers of all groups (group regular, group injected) can be configured but their execution is disabled (limited to rank 1). Refer to function LL_ADC_SetSequencersScanMode().
- Sequencer disabled is equivalent to sequencer of 1 rank: ADC conversion on only 1 channel.

Reference Manual to
LL API cross
reference:

- SQR1 L LL_ADC_REG_SetSequencerLength

LL_ADC_REG_SetSequencerDiscont

Function name

```
STATIC_INLINE void LL_ADC_REG_SetSequencerDiscont  
(ADC_TypeDef * ADCx, uint32_t SeqDiscont)
```

Function description

Set ADC group regular sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks.

Parameters

- **ADCx:** ADC instance
- **SeqDiscont:** This parameter can be one of the following values:
 - LL_ADC_REG_SEQ_DISCONT_DISABLE
 - LL_ADC_REG_SEQ_DISCONT_1RANK
 - LL_ADC_REG_SEQ_DISCONT_2RANKS
 - LL_ADC_REG_SEQ_DISCONT_3RANKS
 - LL_ADC_REG_SEQ_DISCONT_4RANKS
 - LL_ADC_REG_SEQ_DISCONT_5RANKS
 - LL_ADC_REG_SEQ_DISCONT_6RANKS
 - LL_ADC_REG_SEQ_DISCONT_7RANKS
 - LL_ADC_REG_SEQ_DISCONT_8RANKS

Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> It is not possible to enable both ADC group regular continuous mode and sequencer discontinuous mode. It is not possible to enable both ADC auto-injected mode and ADC group regular sequencer discontinuous mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 DISCEN LL_ADC_REG_SetSequencerDiscont CR1 DISCNUM LL_ADC_REG_SetSequencerDiscont

LL_ADC_REG_GetSequencerDiscont

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_REG_GetSequencerDiscont (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_REG_SEQ_DISCONT_DISABLE - LL_ADC_REG_SEQ_DISCONT_1RANK - LL_ADC_REG_SEQ_DISCONT_2RANKS - LL_ADC_REG_SEQ_DISCONT_3RANKS - LL_ADC_REG_SEQ_DISCONT_4RANKS - LL_ADC_REG_SEQ_DISCONT_5RANKS - LL_ADC_REG_SEQ_DISCONT_6RANKS - LL_ADC_REG_SEQ_DISCONT_7RANKS - LL_ADC_REG_SEQ_DISCONT_8RANKS
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 DISCEN LL_ADC_REG_SetSequencerDiscont CR1 DISCNUM LL_ADC_REG_SetSequencerDiscont

LL_ADC_REG_SetSequencerRanks

Function name	<code>__STATIC_INLINE void LL_ADC_REG_SetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank, uint32_t Channel)</code>
Function description	Set ADC group regular sequence: channel on the selected scan sequence rank.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance Rank: This parameter can be one of the following values: (1) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.3, Cat.4 and Cat.5. <ul style="list-style-type: none"> - LL_ADC_REG_RANK_1 - LL_ADC_REG_RANK_2 - LL_ADC_REG_RANK_3 - LL_ADC_REG_RANK_4 - LL_ADC_REG_RANK_5 - LL_ADC_REG_RANK_6 - LL_ADC_REG_RANK_7 - LL_ADC_REG_RANK_8 - LL_ADC_REG_RANK_9

- LL_ADC_REG_RANK_10
 - LL_ADC_REG_RANK_11
 - LL_ADC_REG_RANK_12
 - LL_ADC_REG_RANK_13
 - LL_ADC_REG_RANK_14
 - LL_ADC_REG_RANK_15
 - LL_ADC_REG_RANK_16
 - LL_ADC_REG_RANK_17
 - LL_ADC_REG_RANK_18
 - LL_ADC_REG_RANK_19
 - LL_ADC_REG_RANK_20
 - LL_ADC_REG_RANK_21
 - LL_ADC_REG_RANK_22
 - LL_ADC_REG_RANK_23
 - LL_ADC_REG_RANK_24
 - LL_ADC_REG_RANK_25
 - LL_ADC_REG_RANK_26
 - LL_ADC_REG_RANK_27
 - LL_ADC_REG_RANK_28 (1)
- **Channel:** This parameter can be one of the following values:
(1) On STM32L1, connection via routing interface (RI)
specificity: fast channel (channel routed directly to ADC switch matrix).
 - LL_ADC_CHANNEL_0 (2)
 - LL_ADC_CHANNEL_1 (2)
 - LL_ADC_CHANNEL_2 (2)
 - LL_ADC_CHANNEL_3 (2)
 - LL_ADC_CHANNEL_4 (1)
 - LL_ADC_CHANNEL_5 (1)
 - LL_ADC_CHANNEL_6 (2)
 - LL_ADC_CHANNEL_7 (2)
 - LL_ADC_CHANNEL_8 (2)
 - LL_ADC_CHANNEL_9 (2)
 - LL_ADC_CHANNEL_10 (2)
 - LL_ADC_CHANNEL_11 (2)
 - LL_ADC_CHANNEL_12 (2)
 - LL_ADC_CHANNEL_13 (3)
 - LL_ADC_CHANNEL_14 (3)
 - LL_ADC_CHANNEL_15 (3)
 - LL_ADC_CHANNEL_16 (3)
 - LL_ADC_CHANNEL_17 (3)
 - LL_ADC_CHANNEL_18 (3)
 - LL_ADC_CHANNEL_19 (3)
 - LL_ADC_CHANNEL_20 (3)
 - LL_ADC_CHANNEL_21 (3)
 - LL_ADC_CHANNEL_22 (1)
 - LL_ADC_CHANNEL_23 (1)
 - LL_ADC_CHANNEL_24 (1)
 - LL_ADC_CHANNEL_25 (1)
 - LL_ADC_CHANNEL_26 (3)
 - LL_ADC_CHANNEL_27 (3)(4)
 - LL_ADC_CHANNEL_28 (3)(4)
 - LL_ADC_CHANNEL_29 (3)(4)

- LL_ADC_CHANNEL_30 (3)(4)
 - LL_ADC_CHANNEL_31 (3)(4)
 - LL_ADC_CHANNEL_VREFINT (3)
 - LL_ADC_CHANNEL_TEMPSENSOR (3)
 - LL_ADC_CHANNEL_VCOMP (3)
 - LL_ADC_CHANNEL_VOPAMP1 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP2 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP3 (3)(5)
 - (2) On STM32L1, for devices with feature 'channels banks' available: Channel different in bank A and bank B.
 - (3) On STM32L1, for devices with feature 'channels banks' available: Channel common to both bank A and bank B.
 - (4) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
 - (5) On STM32L1, parameter not available on all devices: OPAMP1 and OPAMP2 available only on STM32L1 Cat.3, Cat.4 and Cat.5, OPAMP3 available only on STM32L1 Cat.4 and Cat.5
- Return values**
- **None:**
- Notes**
- This function performs configuration of: Channels ordering into each rank of scan sequence: whatever channel can be placed into whatever rank.
 - On this STM32 serie, ADC group regular sequencer is fully configurable: sequencer length and each rank affectation to a channel are configurable. Refer to description of function LL_ADC_REG_SetSequencerLength().
 - Depending on devices and packages, some channels may not be available. Refer to device datasheet for channels availability.
 - On this STM32 serie, to measure internal channels (VrefInt, TempSensor, ...), measurement paths to internal channels must be enabled separately. This can be done using function LL_ADC_SetCommonPathInternalCh().
- Reference Manual to LL API cross reference:**
- SQR5 SQ1 LL_ADC_REG_SetSequencerRanks
 - SQR5 SQ2 LL_ADC_REG_SetSequencerRanks
 - SQR5 SQ3 LL_ADC_REG_SetSequencerRanks
 - SQR5 SQ4 LL_ADC_REG_SetSequencerRanks
 - SQR5 SQ5 LL_ADC_REG_SetSequencerRanks
 - SQR5 SQ6 LL_ADC_REG_SetSequencerRanks
 - SQR4 SQ7 LL_ADC_REG_SetSequencerRanks
 - SQR4 SQ8 LL_ADC_REG_SetSequencerRanks
 - SQR4 SQ9 LL_ADC_REG_SetSequencerRanks
 - SQR4 SQ10 LL_ADC_REG_SetSequencerRanks
 - SQR4 SQ11 LL_ADC_REG_SetSequencerRanks
 - SQR4 SQ12 LL_ADC_REG_SetSequencerRanks
 - SQR3 SQ13 LL_ADC_REG_SetSequencerRanks
 - SQR3 SQ14 LL_ADC_REG_SetSequencerRanks
 - SQR3 SQ15 LL_ADC_REG_SetSequencerRanks
 - SQR3 SQ16 LL_ADC_REG_SetSequencerRanks
 - SQR3 SQ17 LL_ADC_REG_SetSequencerRanks
 - SQR3 SQ18 LL_ADC_REG_SetSequencerRanks
 - SQR2 SQ19 LL_ADC_REG_SetSequencerRanks

- SQR2 SQ20 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ21 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ22 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ23 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ24 LL_ADC_REG_SetSequencerRanks
- SQR1 SQ25 LL_ADC_REG_SetSequencerRanks
- SQR1 SQ26 LL_ADC_REG_SetSequencerRanks
- SQR1 SQ27 LL_ADC_REG_SetSequencerRanks
- SQR1 SQ28 LL_ADC_REG_SetSequencerRanks

LL_ADC_REG_GetSequencerRanks

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_REG_GetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank)</code>
Function description	Get ADC group regular sequence: channel on the selected scan sequence rank.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Rank: This parameter can be one of the following values: (1) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.3, Cat.4 and Cat.5. <ul style="list-style-type: none"> – LL_ADC_REG_RANK_1 – LL_ADC_REG_RANK_2 – LL_ADC_REG_RANK_3 – LL_ADC_REG_RANK_4 – LL_ADC_REG_RANK_5 – LL_ADC_REG_RANK_6 – LL_ADC_REG_RANK_7 – LL_ADC_REG_RANK_8 – LL_ADC_REG_RANK_9 – LL_ADC_REG_RANK_10 – LL_ADC_REG_RANK_11 – LL_ADC_REG_RANK_12 – LL_ADC_REG_RANK_13 – LL_ADC_REG_RANK_14 – LL_ADC_REG_RANK_15 – LL_ADC_REG_RANK_16 – LL_ADC_REG_RANK_17 – LL_ADC_REG_RANK_18 – LL_ADC_REG_RANK_19 – LL_ADC_REG_RANK_20 – LL_ADC_REG_RANK_21 – LL_ADC_REG_RANK_22 – LL_ADC_REG_RANK_23 – LL_ADC_REG_RANK_24 – LL_ADC_REG_RANK_25 – LL_ADC_REG_RANK_26 – LL_ADC_REG_RANK_27 – LL_ADC_REG_RANK_28 (1)
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (1) On STM32L1, connection via routing interface (RI) specificity: fast

channel (channel routed directly to ADC switch matrix).

- LL_ADC_CHANNEL_0 (2)
 - LL_ADC_CHANNEL_1 (2)
 - LL_ADC_CHANNEL_2 (2)
 - LL_ADC_CHANNEL_3 (2)
 - LL_ADC_CHANNEL_4 (1)
 - LL_ADC_CHANNEL_5 (1)
 - LL_ADC_CHANNEL_6 (2)
 - LL_ADC_CHANNEL_7 (2)
 - LL_ADC_CHANNEL_8 (2)
 - LL_ADC_CHANNEL_9 (2)
 - LL_ADC_CHANNEL_10 (2)
 - LL_ADC_CHANNEL_11 (2)
 - LL_ADC_CHANNEL_12 (2)
 - LL_ADC_CHANNEL_13 (3)
 - LL_ADC_CHANNEL_14 (3)
 - LL_ADC_CHANNEL_15 (3)
 - LL_ADC_CHANNEL_16 (3)
 - LL_ADC_CHANNEL_17 (3)
 - LL_ADC_CHANNEL_18 (3)
 - LL_ADC_CHANNEL_19 (3)
 - LL_ADC_CHANNEL_20 (3)
 - LL_ADC_CHANNEL_21 (3)
 - LL_ADC_CHANNEL_22 (1)
 - LL_ADC_CHANNEL_23 (1)
 - LL_ADC_CHANNEL_24 (1)
 - LL_ADC_CHANNEL_25 (1)
 - LL_ADC_CHANNEL_26 (3)
 - LL_ADC_CHANNEL_27 (3)(4)
 - LL_ADC_CHANNEL_28 (3)(4)
 - LL_ADC_CHANNEL_29 (3)(4)
 - LL_ADC_CHANNEL_30 (3)(4)
 - LL_ADC_CHANNEL_31 (3)(4)
 - LL_ADC_CHANNEL_VREFINT (3)(6)
 - LL_ADC_CHANNEL_TEMPSENSOR (3)(6)
 - LL_ADC_CHANNEL_VCOMP (3)(6)
 - LL_ADC_CHANNEL_VOPAMP1 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP2 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP3 (3)(5)
- (2) On STM32L1, for devices with feature 'channels banks' available: Channel different in bank A and bank B.
 - (3) On STM32L1, for devices with feature 'channels banks' available: Channel common to both bank A and bank B.
 - (4) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
 - (5) On STM32L1, parameter not available on all devices: OPAMP1 and OPAMP2 available only on STM32L1 Cat.3, Cat.4 and Cat.5, OPAMP3 available only on STM32L1 Cat.4 and Cat.5.
 - (6) For ADC channel read back from ADC register, comparison with internal channel parameter to be done using helper macro
`__LL_ADC_CHANNEL_INTERNAL_TO_EXTERNAL()`.

Notes	<ul style="list-style-type: none"> On this STM32 serie, ADC group regular sequencer is fully configurable: sequencer length and each rank affectation to a channel are configurable. Refer to description of function LL_ADC_REG_SetSequencerLength(). Depending on devices and packages, some channels may not be available. Refer to device datasheet for channels availability. Usage of the returned channel number: To reinject this channel into another function LL_ADC_xxx: the returned channel number is only partly formatted on definition of literals LL_ADC_CHANNEL_x. Therefore, it has to be compared with parts of literals LL_ADC_CHANNEL_x or using helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB(). Then the selected literal LL_ADC_CHANNEL_x can be used as parameter for another function. To get the channel number in decimal format: process the returned value with the helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SQR5 SQ1 LL_ADC_REG_GetSequencerRanks SQR5 SQ2 LL_ADC_REG_GetSequencerRanks SQR5 SQ3 LL_ADC_REG_GetSequencerRanks SQR5 SQ4 LL_ADC_REG_GetSequencerRanks SQR5 SQ5 LL_ADC_REG_GetSequencerRanks SQR5 SQ6 LL_ADC_REG_GetSequencerRanks SQR4 SQ7 LL_ADC_REG_GetSequencerRanks SQR4 SQ8 LL_ADC_REG_GetSequencerRanks SQR4 SQ9 LL_ADC_REG_GetSequencerRanks SQR4 SQ10 LL_ADC_REG_GetSequencerRanks SQR4 SQ11 LL_ADC_REG_GetSequencerRanks SQR4 SQ12 LL_ADC_REG_GetSequencerRanks SQR3 SQ13 LL_ADC_REG_GetSequencerRanks SQR3 SQ14 LL_ADC_REG_GetSequencerRanks SQR3 SQ15 LL_ADC_REG_GetSequencerRanks SQR3 SQ16 LL_ADC_REG_GetSequencerRanks SQR3 SQ17 LL_ADC_REG_GetSequencerRanks SQR3 SQ18 LL_ADC_REG_GetSequencerRanks SQR2 SQ19 LL_ADC_REG_GetSequencerRanks SQR2 SQ20 LL_ADC_REG_GetSequencerRanks SQR2 SQ21 LL_ADC_REG_GetSequencerRanks SQR2 SQ22 LL_ADC_REG_GetSequencerRanks SQR2 SQ23 LL_ADC_REG_GetSequencerRanks SQR2 SQ24 LL_ADC_REG_GetSequencerRanks SQR1 SQ25 LL_ADC_REG_GetSequencerRanks SQR1 SQ26 LL_ADC_REG_GetSequencerRanks SQR1 SQ27 LL_ADC_REG_GetSequencerRanks SQR1 SQ28 LL_ADC_REG_GetSequencerRanks

LL_ADC_REG_SetContinuousMode

Function name	STATIC_INLINE void LL_ADC_REG_SetContinuousMode (ADC_TypeDef * ADCx, uint32_t Continuous)
Function description	Set ADC continuous conversion mode on ADC group regular.

Parameters	<ul style="list-style-type: none"> ADCx: ADC instance Continuous: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_REG_CONV_SINGLE – LL_ADC_REG_CONV_CONTINUOUS
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Description of ADC continuous conversion mode: single mode: one conversion per triggercontinuous mode: after the first trigger, following conversions launched successively automatically. It is not possible to enable both ADC group regular continuous mode and sequencer discontinuous mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 CONT LL_ADC_REG_SetContinuousMode

LL_ADC_REG_GetContinuousMode

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_REG_GetContinuousMode (ADC_TypeDef * ADCx)</code>
Function description	Get ADC continuous conversion mode on ADC group regular.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_REG_CONV_SINGLE – LL_ADC_REG_CONV_CONTINUOUS
Notes	<ul style="list-style-type: none"> Description of ADC continuous conversion mode: single mode: one conversion per triggercontinuous mode: after the first trigger, following conversions launched successively automatically.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 CONT LL_ADC_REG_GetContinuousMode

LL_ADC_REG_SetDMATransfer

Function name	<code>__STATIC_INLINE void LL_ADC_REG_SetDMATransfer (ADC_TypeDef * ADCx, uint32_t DMATransfer)</code>
Function description	Set ADC group regular conversion data transfer: no transfer or transfer by DMA, and DMA requests mode.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance DMATransfer: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_REG_DMA_TRANSFER_NONE – LL_ADC_REG_DMA_TRANSFER_LIMITED – LL_ADC_REG_DMA_TRANSFER_UNLIMITED
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> If transfer by DMA selected, specifies the DMA requests mode: Limited mode (One shot mode): DMA transfer requests

are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular.Unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transfers (number of ADC conversions). This ADC mode is intended to be used with DMA mode circular.

- If ADC DMA requests mode is set to unlimited and DMA is set to mode non-circular: when DMA transfers size will be reached, DMA will stop transfers of ADC conversions data ADC will raise an overrun error (overrun flag and interruption if enabled).
- To configure DMA source address (peripheral address), use function LL_ADC_DMA_GetRegAddr().

Reference Manual to
LL API cross
reference:

- CR2 DMA LL_ADC_REG_SetDMATransfer
- CR2 DDS LL_ADC_REG_SetDMATransfer

LL_ADC_REG_GetDMATransfer

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_REG_GetDMATransfer(ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion data transfer: no transfer or transfer by DMA, and DMA requests mode.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_REG_DMA_TRANSFER_NONE - LL_ADC_REG_DMA_TRANSFER_LIMITED - LL_ADC_REG_DMA_TRANSFER_UNLIMITED
Notes	<ul style="list-style-type: none"> • If transfer by DMA selected, specifies the DMA requests mode: Limited mode (One shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular.Unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transfers (number of ADC conversions). This ADC mode is intended to be used with DMA mode circular. • If ADC DMA requests mode is set to unlimited and DMA is set to mode non-circular: when DMA transfers size will be reached, DMA will stop transfers of ADC conversions data ADC will raise an overrun error (overrun flag and interruption if enabled). • To configure DMA source address (peripheral address), use function LL_ADC_DMA_GetRegAddr().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 DMA LL_ADC_REG_GetDMATransfer • CR2 DDS LL_ADC_REG_GetDMATransfer

LL_ADC_REG_SetFlagEndOfConversion

Function name	<code>_STATIC_INLINE void LL_ADC_REG_SetFlagEndOfConversion (ADC_TypeDef *</code>
---------------	--

ADCx, uint32_t EocSelection)

Function description	Specify which ADC flag between EOC (end of unitary conversion) or EOS (end of sequence conversions) is used to indicate the end of conversion.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • EocSelection: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_REG_FLAG_EOC_SEQUENCE_CONV – LL_ADC_REG_FLAG_EOC_UNITARY_CONV
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This feature is aimed to be set when using ADC with programming model by polling or interruption (programming model by DMA usually uses DMA interruptions to indicate end of conversion and data transfer). • For ADC group injected, end of conversion (flag&IT) is raised only at the end of the sequence.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 EOCS LL_ADC_REG_SetFlagEndOfConversion

LL_ADC_REG_GetFlagEndOfConversion

Function name	STATIC_INLINE uint32_t LL_ADC_REG_GetFlagEndOfConversion (ADC_TypeDef * ADCx)
Function description	Get which ADC flag between EOC (end of unitary conversion) or EOS (end of sequence conversions) is used to indicate the end of conversion.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_REG_FLAG_EOC_SEQUENCE_CONV – LL_ADC_REG_FLAG_EOC_UNITARY_CONV
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 EOCS LL_ADC_REG_GetFlagEndOfConversion

LL_ADC_INJ_SetTriggerSource

Function name	STATIC_INLINE void LL_ADC_INJ_SetTriggerSource (ADC_TypeDef * ADCx, uint32_t TriggerSource)
Function description	Set ADC group injected conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line).
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • TriggerSource: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_INJ_TRIG_SOFTWARE – LL_ADC_INJ_TRIG_EXT_TIM9_CH1 – LL_ADC_INJ_TRIG_EXT_TIM9_TRGO

- LL_ADC_INJ_TRIG_EXT_TIM2_TRGO
- LL_ADC_INJ_TRIG_EXT_TIM2_CH1
- LL_ADC_INJ_TRIG_EXT_TIM3_CH4
- LL_ADC_INJ_TRIG_EXT_TIM4_TRGO
- LL_ADC_INJ_TRIG_EXT_TIM4_CH1
- LL_ADC_INJ_TRIG_EXT_TIM4_CH2
- LL_ADC_INJ_TRIG_EXT_TIM4_CH3
- LL_ADC_INJ_TRIG_EXT_TIM10_CH1
- LL_ADC_INJ_TRIG_EXT_TIM7_TRGO
- LL_ADC_INJ_TRIG_EXT EXTI_LINE15

Return values

- **None:**

Notes

- On this STM32 serie, setting of external trigger edge is performed using function LL_ADC_INJ_StartConversionExtTrig().
- Availability of parameters of trigger sources from timer depends on timers availability on the selected device.

Reference Manual to
LL API cross
reference:

- CR2 JEXTSEL LL_ADC_INJ_SetTriggerSource
- CR2 JEXTEN LL_ADC_INJ_SetTriggerSource

LL_ADC_INJ_GetTriggerSource

Function name

```
_STATIC_INLINE uint32_t LL_ADC_INJ_GetTriggerSource  
(ADC_TypeDef * ADCx)
```

Function description

Get ADC group injected conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line).

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:

- LL_ADC_INJ_TRIG_SOFTWARE
- LL_ADC_INJ_TRIG_EXT_TIM9_CH1
- LL_ADC_INJ_TRIG_EXT_TIM9_TRGO
- LL_ADC_INJ_TRIG_EXT_TIM2_TRGO
- LL_ADC_INJ_TRIG_EXT_TIM2_CH1
- LL_ADC_INJ_TRIG_EXT_TIM3_CH4
- LL_ADC_INJ_TRIG_EXT_TIM4_TRGO
- LL_ADC_INJ_TRIG_EXT_TIM4_CH1
- LL_ADC_INJ_TRIG_EXT_TIM4_CH2
- LL_ADC_INJ_TRIG_EXT_TIM4_CH3
- LL_ADC_INJ_TRIG_EXT_TIM10_CH1
- LL_ADC_INJ_TRIG_EXT_TIM7_TRGO
- LL_ADC_INJ_TRIG_EXT EXTI_LINE15

Notes

- To determine whether group injected trigger source is internal (SW start) or external, without detail of which peripheral is selected as external trigger, (equivalent to "if(LL_ADC_INJ_GetTriggerSource(ADC1) == LL_ADC_INJ_TRIG_SOFTWARE)") use function LL_ADC_INJ_IsTriggerSourceSWStart.
- Availability of parameters of trigger sources from timer depends on timers availability on the selected device.

Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 JEXTSEL LL_ADC_INJ_GetTriggerSource • CR2 JEXTEN LL_ADC_INJ_GetTriggerSource
---	---

LL_ADC_INJ_IsTriggerSourceSWStart

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_INJ_IsTriggerSourceSWStart (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group injected conversion trigger source internal (SW start) or external.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Value: "0" if trigger source external trigger Value "1" if trigger source SW start.
Notes	<ul style="list-style-type: none"> • In case of group injected trigger source set to external trigger, to determine which peripheral is selected as external trigger, use function LL_ADC_INJ_GetTriggerSource.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 JEXTEN LL_ADC_INJ_IsTriggerSourceSWStart

LL_ADC_INJ_GetTriggerEdge

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_INJ_GetTriggerEdge (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group injected conversion trigger polarity.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_INJ_TRIG_EXT_RISING - LL_ADC_INJ_TRIG_EXT_FALLING - LL_ADC_INJ_TRIG_EXT_RISINGFALLING
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 JEXTEN LL_ADC_INJ_GetTriggerEdge

LL_ADC_INJ_SetSequencerLength

Function name	<code>__STATIC_INLINE void LL_ADC_INJ_SetSequencerLength (ADC_TypeDef * ADCx, uint32_t SequencerNbRanks)</code>
Function description	Set ADC group injected sequencer length and scan direction.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • SequencerNbRanks: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_INJ_SEQ_SCAN_DISABLE - LL_ADC_INJ_SEQ_SCAN_ENABLE_2RANKS - LL_ADC_INJ_SEQ_SCAN_ENABLE_3RANKS - LL_ADC_INJ_SEQ_SCAN_ENABLE_4RANKS
Return values	<ul style="list-style-type: none"> • None:

Notes	<ul style="list-style-type: none"> This function performs configuration of: Sequence length: Number of ranks in the scan sequence. Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from rank 1 to rank n). On this STM32 serie, group injected sequencer configuration is conditioned to ADC instance sequencer mode. If ADC instance sequencer mode is disabled, sequencers of all groups (group regular, group injected) can be configured but their execution is disabled (limited to rank 1). Refer to function LL_ADC_SetSequencersScanMode(). Sequencer disabled is equivalent to sequencer of 1 rank: ADC conversion on only 1 channel.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> JSQR JL LL_ADC_INJ_SetSequencerLength

LL_ADC_INJ_GetSequencerLength

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_INJ_GetSequencerLength (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group injected sequencer length and scan direction.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_INJ_SEQ_SCAN_DISABLE - LL_ADC_INJ_SEQ_SCAN_ENABLE_2RANKS - LL_ADC_INJ_SEQ_SCAN_ENABLE_3RANKS - LL_ADC_INJ_SEQ_SCAN_ENABLE_4RANKS
Notes	<ul style="list-style-type: none"> This function retrieves: Sequence length: Number of ranks in the scan sequence. Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from rank 1 to rank n). On this STM32 serie, group injected sequencer configuration is conditioned to ADC instance sequencer mode. If ADC instance sequencer mode is disabled, sequencers of all groups (group regular, group injected) can be configured but their execution is disabled (limited to rank 1). Refer to function LL_ADC_SetSequencersScanMode(). Sequencer disabled is equivalent to sequencer of 1 rank: ADC conversion on only 1 channel.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> JSQR JL LL_ADC_INJ_SetSequencerLength

LL_ADC_INJ_SetSequencerDiscont

Function name	<code>_STATIC_INLINE void LL_ADC_INJ_SetSequencerDiscont (ADC_TypeDef * ADCx, uint32_t SeqDiscont)</code>
Function description	Set ADC group injected sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks.

Parameters	<ul style="list-style-type: none"> ADCx: ADC instance SeqDiscont: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_INJ_SEQ_DISCONT_DISABLE – LL_ADC_INJ_SEQ_DISCONT_1RANK
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> It is not possible to enable both ADC group injected auto-injected mode and sequencer discontinuous mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 DISCEN LL_ADC_INJ_SetSequencerDiscont

LL_ADC_INJ_GetSequencerDiscont

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_INJ_GetSequencerDiscont (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group injected sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_INJ_SEQ_DISCONT_DISABLE – LL_ADC_INJ_SEQ_DISCONT_1RANK
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 DISCEN LL_ADC_REG_GetSequencerDiscont

LL_ADC_INJ_SetSequencerRanks

Function name	<code>__STATIC_INLINE void LL_ADC_INJ_SetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank, uint32_t Channel)</code>
Function description	Set ADC group injected sequence: channel on the selected sequence rank.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance Rank: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_INJ_RANK_1 – LL_ADC_INJ_RANK_2 – LL_ADC_INJ_RANK_3 – LL_ADC_INJ_RANK_4 Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> (1) On STM32L1, connection via routing interface (RI) specificity: fast channel (channel routed directly to ADC switch matrix). <ul style="list-style-type: none"> – LL_ADC_CHANNEL_0 (2) – LL_ADC_CHANNEL_1 (2) – LL_ADC_CHANNEL_2 (2) – LL_ADC_CHANNEL_3 (2) – LL_ADC_CHANNEL_4 (1) – LL_ADC_CHANNEL_5 (1)

- LL_ADC_CHANNEL_6 (2)
 - LL_ADC_CHANNEL_7 (2)
 - LL_ADC_CHANNEL_8 (2)
 - LL_ADC_CHANNEL_9 (2)
 - LL_ADC_CHANNEL_10 (2)
 - LL_ADC_CHANNEL_11 (2)
 - LL_ADC_CHANNEL_12 (2)
 - LL_ADC_CHANNEL_13 (3)
 - LL_ADC_CHANNEL_14 (3)
 - LL_ADC_CHANNEL_15 (3)
 - LL_ADC_CHANNEL_16 (3)
 - LL_ADC_CHANNEL_17 (3)
 - LL_ADC_CHANNEL_18 (3)
 - LL_ADC_CHANNEL_19 (3)
 - LL_ADC_CHANNEL_20 (3)
 - LL_ADC_CHANNEL_21 (3)
 - LL_ADC_CHANNEL_22 (1)
 - LL_ADC_CHANNEL_23 (1)
 - LL_ADC_CHANNEL_24 (1)
 - LL_ADC_CHANNEL_25 (1)
 - LL_ADC_CHANNEL_26 (3)
 - LL_ADC_CHANNEL_27 (3)(4)
 - LL_ADC_CHANNEL_28 (3)(4)
 - LL_ADC_CHANNEL_29 (3)(4)
 - LL_ADC_CHANNEL_30 (3)(4)
 - LL_ADC_CHANNEL_31 (3)(4)
 - LL_ADC_CHANNEL_VREFINT (3)
 - LL_ADC_CHANNEL_TEMPSENSOR (3)
 - LL_ADC_CHANNEL_VCOMP (3)
 - LL_ADC_CHANNEL_VOPAMP1 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP2 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP3 (3)(5)
- (2) On STM32L1, for devices with feature 'channels banks' available: Channel different in bank A and bank B.
 - (3) On STM32L1, for devices with feature 'channels banks' available: Channel common to both bank A and bank B.
 - (4) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
 - (5) On STM32L1, parameter not available on all devices: OPAMP1 and OPAMP2 available only on STM32L1 Cat.3, Cat.4 and Cat.5, OPAMP3 available only on STM32L1 Cat.4 and Cat.5

Return values

- **None:**

Notes

- Depending on devices and packages, some channels may not be available. Refer to device datasheet for channels availability.
- On this STM32 serie, to measure internal channels (VrefInt, TempSensor, ...), measurement paths to internal channels must be enabled separately. This can be done using function LL_ADC_SetCommonPathInternalCh().

**Reference Manual to
LL API cross**

- JSQR JSQ1 LL_ADC_INJ_SetSequencerRanks
- JSQR JSQ2 LL_ADC_INJ_SetSequencerRanks

- reference:
- JSQR JSQ3 LL_ADC_INJ_SetSequencerRanks
 - JSQR JSQ4 LL_ADC_INJ_SetSequencerRanks

LL_ADC_INJ_GetSequencerRanks

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_INJ_GetSequencerRanks(ADC_TypeDef * ADCx, uint32_t Rank)</code>
Function description	Get ADC group injected sequence: channel on the selected sequence rank.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Rank: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_INJ_RANK_1 - LL_ADC_INJ_RANK_2 - LL_ADC_INJ_RANK_3 - LL_ADC_INJ_RANK_4
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (1) On STM32L1, connection via routing interface (RI) specificity: fast channel (channel routed directly to ADC switch matrix). <ul style="list-style-type: none"> - LL_ADC_CHANNEL_0 (2) - LL_ADC_CHANNEL_1 (2) - LL_ADC_CHANNEL_2 (2) - LL_ADC_CHANNEL_3 (2) - LL_ADC_CHANNEL_4 (1) - LL_ADC_CHANNEL_5 (1) - LL_ADC_CHANNEL_6 (2) - LL_ADC_CHANNEL_7 (2) - LL_ADC_CHANNEL_8 (2) - LL_ADC_CHANNEL_9 (2) - LL_ADC_CHANNEL_10 (2) - LL_ADC_CHANNEL_11 (2) - LL_ADC_CHANNEL_12 (2) - LL_ADC_CHANNEL_13 (3) - LL_ADC_CHANNEL_14 (3) - LL_ADC_CHANNEL_15 (3) - LL_ADC_CHANNEL_16 (3) - LL_ADC_CHANNEL_17 (3) - LL_ADC_CHANNEL_18 (3) - LL_ADC_CHANNEL_19 (3) - LL_ADC_CHANNEL_20 (3) - LL_ADC_CHANNEL_21 (3) - LL_ADC_CHANNEL_22 (1) - LL_ADC_CHANNEL_23 (1) - LL_ADC_CHANNEL_24 (1) - LL_ADC_CHANNEL_25 (1) - LL_ADC_CHANNEL_26 (3) - LL_ADC_CHANNEL_27 (3)(4) - LL_ADC_CHANNEL_28 (3)(4) - LL_ADC_CHANNEL_29 (3)(4) - LL_ADC_CHANNEL_30 (3)(4) - LL_ADC_CHANNEL_31 (3)(4) - LL_ADC_CHANNEL_VREFINT (3)(6)

- LL_ADC_CHANNEL_TEMPSENSOR (3)(6)
 - LL_ADC_CHANNEL_VCOMP (3)(6)
 - LL_ADC_CHANNEL_VOPAMP1 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP2 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP3 (3)(5)
 - (2) On STM32L1, for devices with feature 'channels banks' available: Channel different in bank A and bank B.
 - (3) On STM32L1, for devices with feature 'channels banks' available: Channel common to both bank A and bank B.
 - (4) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
 - (5) On STM32L1, parameter not available on all devices: OPAMP1 and OPAMP2 available only on STM32L1 Cat.3, Cat.4 and Cat.5, OPAMP3 available only on STM32L1 Cat.4 and Cat.5.
 - (6) For ADC channel read back from ADC register, comparison with internal channel parameter to be done using helper macro
`__LL_ADC_CHANNEL_INTERNAL_TO_EXTERNAL()`.
- Notes**
- Depending on devices and packages, some channels may not be available. Refer to device datasheet for channels availability.
 - Usage of the returned channel number: To reinject this channel into another function LL_ADC_xxx: the returned channel number is only partly formatted on definition of literals LL_ADC_CHANNEL_x. Therefore, it has to be compared with parts of literals LL_ADC_CHANNEL_x or using helper macro
`__LL_ADC_CHANNEL_TO_DECIMAL_NB()`. Then the selected literal LL_ADC_CHANNEL_x can be used as parameter for another function. To get the channel number in decimal format: process the returned value with the helper macro
`__LL_ADC_CHANNEL_TO_DECIMAL_NB()`.
- Reference Manual to LL API cross reference:**
- JSQR JSQ1 LL_ADC_INJ_SetSequencerRanks
 - JSQR JSQ2 LL_ADC_INJ_SetSequencerRanks
 - JSQR JSQ3 LL_ADC_INJ_SetSequencerRanks
 - JSQR JSQ4 LL_ADC_INJ_SetSequencerRanks

LL_ADC_INJ_SetTrigAuto

Function name	<code>__STATIC_INLINE void LL_ADC_INJ_SetTrigAuto(ADC_TypeDef * ADCx, uint32_t TrigAuto)</code>
Function description	Set ADC group injected conversion trigger: independent or from ADC group regular.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • TrigAuto: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_INJ_TRIG_INDEPENDENT - LL_ADC_INJ_TRIG_FROM_GRP_REGULAR
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This mode can be used to extend number of data registers updated after one ADC conversion trigger and with data permanently kept (not erased by successive conversions of

	scan of ADC sequencer ranks), up to 5 data registers: 1 data register on ADC group regular, 4 data registers on ADC group injected.
	<ul style="list-style-type: none"> • If ADC group injected injected trigger source is set to an external trigger, this feature must be must be set to independent trigger. ADC group injected automatic trigger is compliant only with group injected trigger source set to SW start, without any further action on ADC group injected conversion start or stop: in this case, ADC group injected is controlled only from ADC group regular. • It is not possible to enable both ADC group injected auto-injected mode and sequencer discontinuous mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 JAUTO LL_ADC_INJ_SetTrigAuto

LL_ADC_INJ_GetTrigAuto

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_INJ_GetTrigAuto (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group injected conversion trigger: independent or from ADC group regular.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - <code>LL_ADC_INJ_TRIG_INDEPENDENT</code> - <code>LL_ADC_INJ_TRIG_FROM_GRP_REGULAR</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 JAUTO LL_ADC_INJ_GetTrigAuto

LL_ADC_INJ_SetOffset

Function name	<code>_STATIC_INLINE void LL_ADC_INJ_SetOffset (ADC_TypeDef * ADCx, uint32_t Rank, uint32_t OffsetLevel)</code>
Function description	Set ADC group injected offset.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Rank: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_ADC_INJ_RANK_1</code> - <code>LL_ADC_INJ_RANK_2</code> - <code>LL_ADC_INJ_RANK_3</code> - <code>LL_ADC_INJ_RANK_4</code> • OffsetLevel: Value between Min_Data=0x000 and Max_Data=0xFFFF
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • It sets: ADC group injected rank to which the offset programmed will be appliedOffset level (offset to be subtracted from the raw converted data). Caution: Offset format is dependent to ADC resolution: offset has to be left-aligned on bit 11, the LSB (right bits) are set to 0. • Offset cannot be enabled or disabled. To emulate offset

disabled, set an offset value equal to 0.

Reference Manual to
LL API cross
reference:

- JOFR1 JOFFSET1 LL_ADC_INJ_SetOffset
- JOFR2 JOFFSET2 LL_ADC_INJ_SetOffset
- JOFR3 JOFFSET3 LL_ADC_INJ_SetOffset
- JOFR4 JOFFSET4 LL_ADC_INJ_SetOffset

LL_ADC_INJ_GetOffset

Function name

**_STATIC_INLINE uint32_t LL_ADC_INJ_GetOffset
(ADC_TypeDef * ADCx, uint32_t Rank)**

Function description

Get ADC group injected offset.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
 - LL_ADC_INJ_RANK_1
 - LL_ADC_INJ_RANK_2
 - LL_ADC_INJ_RANK_3
 - LL_ADC_INJ_RANK_4

Return values

- **Value:** between Min_Data=0x000 and Max_Data=0xFFFF

Notes

- It gives offset level (offset to be subtracted from the raw converted data). Caution: Offset format is dependent to ADC resolution: offset has to be left-aligned on bit 11, the LSB (right bits) are set to 0.

Reference Manual to
LL API cross
reference:

- JOFR1 JOFFSET1 LL_ADC_INJ_SetOffset
- JOFR2 JOFFSET2 LL_ADC_INJ_SetOffset
- JOFR3 JOFFSET3 LL_ADC_INJ_SetOffset
- JOFR4 JOFFSET4 LL_ADC_INJ_SetOffset

LL_ADC_SetChannelSamplingTime

Function name

**_STATIC_INLINE void LL_ADC_SetChannelSamplingTime
(ADC_TypeDef * ADCx, uint32_t Channel, uint32_t
SamplingTime)**

Function description

Set sampling time of the selected ADC channel Unit: ADC clock cycles.

Parameters

- **ADCx:** ADC instance
- **Channel:** This parameter can be one of the following values:
 - (1) On STM32L1, connection via routing interface (RI) specificity: fast channel (channel routed directly to ADC switch matrix).
 - LL_ADC_CHANNEL_0 (2)
 - LL_ADC_CHANNEL_1 (2)
 - LL_ADC_CHANNEL_2 (2)
 - LL_ADC_CHANNEL_3 (2)
 - LL_ADC_CHANNEL_4 (1)
 - LL_ADC_CHANNEL_5 (1)
 - LL_ADC_CHANNEL_6 (2)
 - LL_ADC_CHANNEL_7 (2)
 - LL_ADC_CHANNEL_8 (2)
 - LL_ADC_CHANNEL_9 (2)

- LL_ADC_CHANNEL_10 (2)
- LL_ADC_CHANNEL_11 (2)
- LL_ADC_CHANNEL_12 (2)
- LL_ADC_CHANNEL_13 (3)
- LL_ADC_CHANNEL_14 (3)
- LL_ADC_CHANNEL_15 (3)
- LL_ADC_CHANNEL_16 (3)
- LL_ADC_CHANNEL_17 (3)
- LL_ADC_CHANNEL_18 (3)
- LL_ADC_CHANNEL_19 (3)
- LL_ADC_CHANNEL_20 (3)
- LL_ADC_CHANNEL_21 (3)
- LL_ADC_CHANNEL_22 (1)
- LL_ADC_CHANNEL_23 (1)
- LL_ADC_CHANNEL_24 (1)
- LL_ADC_CHANNEL_25 (1)
- LL_ADC_CHANNEL_26 (3)
- LL_ADC_CHANNEL_27 (3)(4)
- LL_ADC_CHANNEL_28 (3)(4)
- LL_ADC_CHANNEL_29 (3)(4)
- LL_ADC_CHANNEL_30 (3)(4)
- LL_ADC_CHANNEL_31 (3)(4)
- LL_ADC_CHANNEL_VREFINT (3)
- LL_ADC_CHANNEL_TEMPSENSOR (3)
- LL_ADC_CHANNEL_VCOMP (3)
- LL_ADC_CHANNEL_VOPAMP1 (3)(5)
- LL_ADC_CHANNEL_VOPAMP2 (3)(5)
- LL_ADC_CHANNEL_VOPAMP3 (3)(5)
- (2) On STM32L1, for devices with feature 'channels banks' available: Channel different in bank A and bank B.
- (3) On STM32L1, for devices with feature 'channels banks' available: Channel common to both bank A and bank B.
- (4) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
- (5) On STM32L1, parameter not available on all devices: OPAMP1 and OPAMP2 available only on STM32L1 Cat.3, Cat.4 and Cat.5, OPAMP3 available only on STM32L1 Cat.4 and Cat.5
- **SamplingTime:** This parameter can be one of the following values:
 - LL_ADC_SAMPLINGTIME_4CYCLES
 - LL_ADC_SAMPLINGTIME_9CYCLES
 - LL_ADC_SAMPLINGTIME_16CYCLES
 - LL_ADC_SAMPLINGTIME_24CYCLES
 - LL_ADC_SAMPLINGTIME_48CYCLES
 - LL_ADC_SAMPLINGTIME_96CYCLES
 - LL_ADC_SAMPLINGTIME_192CYCLES
 - LL_ADC_SAMPLINGTIME_384CYCLES

Return values

- **None:**

Notes

- On this device, sampling time is on channel scope: independently of channel mapped on ADC group regular or injected.

- In case of internal channel (VrefInt, TempSensor, ...) to be converted: sampling time constraints must be respected (sampling time can be adjusted in function of ADC clock frequency and sampling time setting). Refer to device datasheet for timings values (parameters TS_vrefint, TS_temp, ...).
- Conversion time is the addition of sampling time and processing time. Refer to reference manual for ADC processing time of this STM32 serie.
- In case of ADC conversion of internal channel (VrefInt, temperature sensor, ...), a sampling time minimum value is required. Refer to device datasheet.

Reference Manual to
LL API cross
reference:

- SMPR0 SMP31 LL_ADC_SetChannelSamplingTime
- SMPR0 SMP30 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP29 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP28 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP27 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP26 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP25 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP24 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP23 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP22 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP21 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP20 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP19 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP18 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP17 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP16 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP15 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP14 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP13 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP12 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP11 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP10 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP9 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP8 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP7 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP6 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP5 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP4 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP3 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP2 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP1 LL_ADC_SetChannelSamplingTime
- SMPR3 SMP0 LL_ADC_SetChannelSamplingTime

LL_ADC_GetChannelSamplingTime

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_GetChannelSamplingTime (ADC_TypeDef * ADCx, uint32_t Channel)</code>
Function description	Get sampling time of the selected ADC channel Unit: ADC clock cycles.

Parameters

- **ADCx:** ADC instance
- **Channel:** This parameter can be one of the following values:
 - (1) On STM32L1, connection via routing interface (RI) specificity: fast channel (channel routed directly to ADC switch matrix).
 - LL_ADC_CHANNEL_0 (2)
 - LL_ADC_CHANNEL_1 (2)
 - LL_ADC_CHANNEL_2 (2)
 - LL_ADC_CHANNEL_3 (2)
 - LL_ADC_CHANNEL_4 (1)
 - LL_ADC_CHANNEL_5 (1)
 - LL_ADC_CHANNEL_6 (2)
 - LL_ADC_CHANNEL_7 (2)
 - LL_ADC_CHANNEL_8 (2)
 - LL_ADC_CHANNEL_9 (2)
 - LL_ADC_CHANNEL_10 (2)
 - LL_ADC_CHANNEL_11 (2)
 - LL_ADC_CHANNEL_12 (2)
 - LL_ADC_CHANNEL_13 (3)
 - LL_ADC_CHANNEL_14 (3)
 - LL_ADC_CHANNEL_15 (3)
 - LL_ADC_CHANNEL_16 (3)
 - LL_ADC_CHANNEL_17 (3)
 - LL_ADC_CHANNEL_18 (3)
 - LL_ADC_CHANNEL_19 (3)
 - LL_ADC_CHANNEL_20 (3)
 - LL_ADC_CHANNEL_21 (3)
 - LL_ADC_CHANNEL_22 (1)
 - LL_ADC_CHANNEL_23 (1)
 - LL_ADC_CHANNEL_24 (1)
 - LL_ADC_CHANNEL_25 (1)
 - LL_ADC_CHANNEL_26 (3)
 - LL_ADC_CHANNEL_27 (3)(4)
 - LL_ADC_CHANNEL_28 (3)(4)
 - LL_ADC_CHANNEL_29 (3)(4)
 - LL_ADC_CHANNEL_30 (3)(4)
 - LL_ADC_CHANNEL_31 (3)(4)
 - LL_ADC_CHANNEL_VREFINT (3)
 - LL_ADC_CHANNEL_TEMPSENSOR (3)
 - LL_ADC_CHANNEL_VCOMP (3)
 - LL_ADC_CHANNEL_VOPAMP1 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP2 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP3 (3)(5)
 - (2) On STM32L1, for devices with feature 'channels banks' available: Channel different in bank A and bank B.
 - (3) On STM32L1, for devices with feature 'channels banks' available: Channel common to both bank A and bank B.
 - (4) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
 - (5) On STM32L1, parameter not available on all devices: OPAMP1 and OPAMP2 available only on STM32L1 Cat.3, Cat.4 and Cat.5, OPAMP3 available only on STM32L1 Cat.4

and Cat.5

- | | |
|---|---|
| Return values | <ul style="list-style-type: none">• Returned: value can be one of the following values:<ul style="list-style-type: none">- LL_ADC_SAMPLINGTIME_4CYCLES- LL_ADC_SAMPLINGTIME_9CYCLES- LL_ADC_SAMPLINGTIME_16CYCLES- LL_ADC_SAMPLINGTIME_24CYCLES- LL_ADC_SAMPLINGTIME_48CYCLES- LL_ADC_SAMPLINGTIME_96CYCLES- LL_ADC_SAMPLINGTIME_192CYCLES- LL_ADC_SAMPLINGTIME_384CYCLES |
| Notes | <ul style="list-style-type: none">• On this device, sampling time is on channel scope: independently of channel mapped on ADC group regular or injected.• Conversion time is the addition of sampling time and processing time. Refer to reference manual for ADC processing time of this STM32 serie. |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none">• SMPR0 SMP31 LL_ADC_GetChannelSamplingTime• SMPR0 SMP30 LL_ADC_GetChannelSamplingTime• SMPR1 SMP29 LL_ADC_GetChannelSamplingTime• SMPR1 SMP28 LL_ADC_GetChannelSamplingTime• SMPR1 SMP27 LL_ADC_GetChannelSamplingTime• SMPR1 SMP26 LL_ADC_GetChannelSamplingTime• SMPR1 SMP25 LL_ADC_GetChannelSamplingTime• SMPR1 SMP24 LL_ADC_GetChannelSamplingTime• SMPR1 SMP23 LL_ADC_GetChannelSamplingTime• SMPR1 SMP22 LL_ADC_GetChannelSamplingTime• SMPR1 SMP21 LL_ADC_GetChannelSamplingTime• SMPR1 SMP20 LL_ADC_GetChannelSamplingTime• SMPR2 SMP19 LL_ADC_GetChannelSamplingTime• SMPR2 SMP18 LL_ADC_GetChannelSamplingTime• SMPR2 SMP17 LL_ADC_GetChannelSamplingTime• SMPR2 SMP16 LL_ADC_GetChannelSamplingTime• SMPR2 SMP15 LL_ADC_GetChannelSamplingTime• SMPR2 SMP14 LL_ADC_GetChannelSamplingTime• SMPR2 SMP13 LL_ADC_GetChannelSamplingTime• SMPR2 SMP12 LL_ADC_GetChannelSamplingTime• SMPR2 SMP11 LL_ADC_GetChannelSamplingTime• SMPR2 SMP10 LL_ADC_GetChannelSamplingTime• SMPR3 SMP9 LL_ADC_GetChannelSamplingTime• SMPR3 SMP8 LL_ADC_GetChannelSamplingTime• SMPR3 SMP7 LL_ADC_GetChannelSamplingTime• SMPR3 SMP6 LL_ADC_GetChannelSamplingTime• SMPR3 SMP5 LL_ADC_GetChannelSamplingTime• SMPR3 SMP4 LL_ADC_GetChannelSamplingTime• SMPR3 SMP3 LL_ADC_GetChannelSamplingTime• SMPR3 SMP2 LL_ADC_GetChannelSamplingTime• SMPR3 SMP1 LL_ADC_GetChannelSamplingTime• SMPR3 SMP0 LL_ADC_GetChannelSamplingTime |

LL_ADC_SetChannelRouting

Function name	<code>_STATIC_INLINE void LL_ADC_SetChannelRouting (ADC_TypeDef * ADCx, uint32_t Channel, uint32_t Routing)</code>
Function description	Set ADC channels routing.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> (1) Used as ADC direct channel (fast channel) if OPAMP1 is in power down mode. <ul style="list-style-type: none"> – LL_ADC_CHANNEL_3_ROUTING (1) – LL_ADC_CHANNEL_8_ROUTING (2) – LL_ADC_CHANNEL_13_ROUTING (3) (2) Used as ADC direct channel (fast channel) if OPAMP2 is in power down mode. (3) Used as ADC re-routed channel if OPAMP3 is in power down mode. Otherwise, channel 13 is connected to OPAMP3 output and routed through switches COMP1_SW1 and VCOMP to ADC switch matrix. (Note: OPAMP3 is available on STM32L1 Cat.4 only). • Routing: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_CHANNEL_ROUTING_DEFAULT – LL_ADC_CHANNEL_ROUTING_DIRECT
Notes	<ul style="list-style-type: none"> • Channel routing set configuration between ADC IP and GPIO pads, it is used to increase ADC channels speed (setting of direct channel). • This feature is specific to STM32L1, on devices category Cat.3, Cat.4, Cat.5. To use this function, COMP RCC clock domain must be enabled. Refer to LL_APB1_GRP1_PERIPH_COMP.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR FCH3 LL_ADC_SetChannelRouting • CSR FCH8 LL_ADC_SetChannelRouting • CSR RCH13 LL_ADC_SetChannelRouting

LL_ADC_GetChannelRouting

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_GetChannelRouting (ADC_TypeDef * ADCx, uint32_t Channel)</code>
Function description	Get ADC channels speed.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> (1) Used as ADC direct channel (fast channel) if OPAMP1 is in power down mode. <ul style="list-style-type: none"> – LL_ADC_CHANNEL_3_ROUTING (1) – LL_ADC_CHANNEL_8_ROUTING (2) – LL_ADC_CHANNEL_13_ROUTING (3) (2) Used as ADC direct channel (fast channel) if OPAMP2 is in power down mode. (3) Used as ADC re-routed channel if OPAMP3 is in power down mode. Otherwise, channel 13 is connected to OPAMP3 output and routed through switches COMP1_SW1 and VCOMP to ADC switch matrix. (Note: OPAMP3 is available

on STM32L1 Cat.4 only).

Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_ADC_CHANNEL_ROUTING_DEFAULT – LL_ADC_CHANNEL_ROUTING_DIRECT
Notes	<ul style="list-style-type: none"> • Channel routing set configuration between ADC IP and GPIO pads, it is used to increase ADC channels speed (setting of direct channel). • This feature is specific to STM32L1, on devices category Cat.3, Cat.4, Cat.5. To use this function, COMP RCC clock domain must be enabled. Refer to LL_APB1_GRP1_PERIPH_COMP.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR FCH3 LL_ADC_GetChannelRouting • CSR FCH8 LL_ADC_GetChannelRouting • CSR RCH13 LL_ADC_GetChannelRouting

LL_ADC_SetAnalogWDMonitChannels

Function name	<code>_STATIC_INLINE void LL_ADC_SetAnalogWDMonitChannels(ADC_TypeDef * ADCx, uint32_t AWDChannelGroup)</code>
Function description	Set ADC analog watchdog monitored channels: a single channel or all channels, on ADC groups regular and-or injected.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • AWDChannelGroup: This parameter can be one of the following values: (1) On STM32L1, connection via routing interface (RI) specificity: fast channel (channel routed directly to ADC switch matrix). <ul style="list-style-type: none"> – LL_ADC_AWD_DISABLE – LL_ADC_AWD_ALL_CHANNELS_REG – LL_ADC_AWD_ALL_CHANNELS_INJ – LL_ADC_AWD_ALL_CHANNELS_REG_INJ – LL_ADC_AWD_CHANNEL_0_REG (2) – LL_ADC_AWD_CHANNEL_0_INJ (2) – LL_ADC_AWD_CHANNEL_0_REG_INJ (2) – LL_ADC_AWD_CHANNEL_1_REG (2) – LL_ADC_AWD_CHANNEL_1_INJ (2) – LL_ADC_AWD_CHANNEL_1_REG_INJ (2) – LL_ADC_AWD_CHANNEL_2_REG (2) – LL_ADC_AWD_CHANNEL_2_INJ (2) – LL_ADC_AWD_CHANNEL_2_REG_INJ (2) – LL_ADC_AWD_CHANNEL_3_REG (2) – LL_ADC_AWD_CHANNEL_3_INJ (2) – LL_ADC_AWD_CHANNEL_3_REG_INJ (2) – LL_ADC_AWD_CHANNEL_4_REG (1) – LL_ADC_AWD_CHANNEL_4_INJ (1) – LL_ADC_AWD_CHANNEL_4_REG_INJ (1) – LL_ADC_AWD_CHANNEL_5_REG (1) – LL_ADC_AWD_CHANNEL_5_INJ (1) – LL_ADC_AWD_CHANNEL_5_REG_INJ (1) – LL_ADC_AWD_CHANNEL_6_REG (2) – LL_ADC_AWD_CHANNEL_6_INJ (2) – LL_ADC_AWD_CHANNEL_6_REG_INJ (2)

- LL_ADC_AWD_CHANNEL_7_REG (2)
- LL_ADC_AWD_CHANNEL_7_INJ (2)
- LL_ADC_AWD_CHANNEL_7_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_8_REG (2)
- LL_ADC_AWD_CHANNEL_8_INJ (2)
- LL_ADC_AWD_CHANNEL_8_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_9_REG (2)
- LL_ADC_AWD_CHANNEL_9_INJ (2)
- LL_ADC_AWD_CHANNEL_9_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_10_REG (2)
- LL_ADC_AWD_CHANNEL_10_INJ (2)
- LL_ADC_AWD_CHANNEL_10_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_11_REG (2)
- LL_ADC_AWD_CHANNEL_11_INJ (2)
- LL_ADC_AWD_CHANNEL_11_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_12_REG (2)
- LL_ADC_AWD_CHANNEL_12_INJ (2)
- LL_ADC_AWD_CHANNEL_12_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_13_REG (3)
- LL_ADC_AWD_CHANNEL_13_INJ (3)
- LL_ADC_AWD_CHANNEL_13_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_14_REG (3)
- LL_ADC_AWD_CHANNEL_14_INJ (3)
- LL_ADC_AWD_CHANNEL_14_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_15_REG (3)
- LL_ADC_AWD_CHANNEL_15_INJ (3)
- LL_ADC_AWD_CHANNEL_15_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_16_REG (3)
- LL_ADC_AWD_CHANNEL_16_INJ (3)
- LL_ADC_AWD_CHANNEL_16_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_17_REG (3)
- LL_ADC_AWD_CHANNEL_17_INJ (3)
- LL_ADC_AWD_CHANNEL_17_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_18_REG (3)
- LL_ADC_AWD_CHANNEL_18_INJ (3)
- LL_ADC_AWD_CHANNEL_18_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_19_REG (3)
- LL_ADC_AWD_CHANNEL_19_INJ (3)
- LL_ADC_AWD_CHANNEL_19_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_20_REG (3)
- LL_ADC_AWD_CHANNEL_20_INJ (3)
- LL_ADC_AWD_CHANNEL_20_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_21_REG (3)
- LL_ADC_AWD_CHANNEL_21_INJ (3)
- LL_ADC_AWD_CHANNEL_21_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_22_REG (1)
- LL_ADC_AWD_CHANNEL_22_INJ (1)
- LL_ADC_AWD_CHANNEL_22_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_23_REG (1)
- LL_ADC_AWD_CHANNEL_23_INJ (1)
- LL_ADC_AWD_CHANNEL_23_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_24_REG (1)

- LL_ADC_AWD_CHANNEL_24_INJ (1)
 - LL_ADC_AWD_CHANNEL_24_REG_INJ (1)
 - LL_ADC_AWD_CHANNEL_25_REG (1)
 - LL_ADC_AWD_CHANNEL_25_INJ (1)
 - LL_ADC_AWD_CHANNEL_25_REG_INJ (1)
 - LL_ADC_AWD_CHANNEL_26_REG (3)
 - LL_ADC_AWD_CHANNEL_26_INJ (3)
 - LL_ADC_AWD_CHANNEL_26_REG_INJ (3)
 - LL_ADC_AWD_CHANNEL_27_REG (3)(4)
 - LL_ADC_AWD_CHANNEL_27_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_27_REG_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_28_REG (3)(4)
 - LL_ADC_AWD_CHANNEL_28_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_28_REG_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_29_REG (3)(4)
 - LL_ADC_AWD_CHANNEL_29_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_29_REG_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_30_REG (3)(4)
 - LL_ADC_AWD_CHANNEL_30_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_30_REG_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_31_REG (3)(4)
 - LL_ADC_AWD_CHANNEL_31_INJ (3)(4)
 - LL_ADC_AWD_CHANNEL_31_REG_INJ (3)(4)
 - LL_ADC_AWD_CH_VREFINT_REG (3)
 - LL_ADC_AWD_CH_VREFINT_INJ (3)
 - LL_ADC_AWD_CH_VREFINT_REG_INJ (3)
 - LL_ADC_AWD_CH_TEMPSENSOR_REG (3)
 - LL_ADC_AWD_CH_TEMPSENSOR_INJ (3)
 - LL_ADC_AWD_CH_TEMPSENSOR_REG_INJ (3)
 - LL_ADC_AWD_CH_VCOMP_REG (3)
 - LL_ADC_AWD_CH_VCOMP_INJ (3)
 - LL_ADC_AWD_CH_VCOMP_REG_INJ (3)
 - LL_ADC_AWD_CH_VOPAMP1_REG (3)(5)
 - LL_ADC_AWD_CH_VOPAMP1_INJ (3)(5)
 - LL_ADC_AWD_CH_VOPAMP1_REG_INJ (3)(5)
 - LL_ADC_AWD_CH_VOPAMP2_REG (3)(5)
 - LL_ADC_AWD_CH_VOPAMP2_INJ (3)(5)
 - LL_ADC_AWD_CH_VOPAMP2_REG_INJ (3)(5)
 - LL_ADC_AWD_CH_VOPAMP3_REG (3)(5)
 - LL_ADC_AWD_CH_VOPAMP3_INJ (3)(5)
 - LL_ADC_AWD_CH_VOPAMP3_REG_INJ (3)(5)
 - (2) On STM32L1, for devices with feature 'channels banks' available: Channel different in bank A and bank B.
 - (3) On STM32L1, for devices with feature 'channels banks' available: Channel common to both bank A and bank B.
 - (4) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
 - (5) On STM32L1, parameter not available on all devices: OPAMP1 and OPAMP2 available only on STM32L1 Cat.3, Cat.4 and Cat.5, OPAMP3 available only on STM32L1 Cat.4 and Cat.5
- **None:**

Return values

Notes	<ul style="list-style-type: none"> Once monitored channels are selected, analog watchdog is enabled. In case of need to define a single channel to monitor with analog watchdog from sequencer channel definition, use helper macro <code>__LL_ADC_ANALOGWD_CHANNEL_GROUP()</code>. On this STM32 serie, there is only 1 kind of analog watchdog instance: AWD standard (instance AWD1): channels monitored: can monitor 1 channel or all channels.groups monitored: ADC groups regular and-or injected.resolution: resolution is not limited (corresponds to ADC resolution configured).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 AWD1CH LL_ADC_SetAnalogWDMonitChannels CR1 AWD1SGL LL_ADC_SetAnalogWDMonitChannels CR1 AWD1EN LL_ADC_SetAnalogWDMonitChannels

LL_ADC_GetAnalogWDMonitChannels

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_GetAnalogWDMonitChannels (ADC_TypeDef * ADCx)</code>
Function description	Get ADC analog watchdog monitored channel.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: (1) On STM32L1, connection via routing interface (RI) specificity: fast channel (channel routed directly to ADC switch matrix). <ul style="list-style-type: none"> - <code>LL_ADC_AWD_DISABLE</code> - <code>LL_ADC_AWD_ALL_CHANNELS_REG</code> - <code>LL_ADC_AWD_ALL_CHANNELS_INJ</code> - <code>LL_ADC_AWD_ALL_CHANNELS_REG_INJ</code> - <code>LL_ADC_AWD_CHANNEL_0_REG (2)</code> - <code>LL_ADC_AWD_CHANNEL_0_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_0_REG_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_1_REG (2)</code> - <code>LL_ADC_AWD_CHANNEL_1_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_1_REG_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_2_REG (2)</code> - <code>LL_ADC_AWD_CHANNEL_2_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_2_REG_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_3_REG (2)</code> - <code>LL_ADC_AWD_CHANNEL_3_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_3_REG_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_4_REG (1)</code> - <code>LL_ADC_AWD_CHANNEL_4_INJ (1)</code> - <code>LL_ADC_AWD_CHANNEL_4_REG_INJ (1)</code> - <code>LL_ADC_AWD_CHANNEL_5_REG (1)</code> - <code>LL_ADC_AWD_CHANNEL_5_INJ (1)</code> - <code>LL_ADC_AWD_CHANNEL_5_REG_INJ (1)</code> - <code>LL_ADC_AWD_CHANNEL_6_REG (2)</code> - <code>LL_ADC_AWD_CHANNEL_6_INJ (2)</code> - <code>LL_ADC_AWD_CHANNEL_6_REG_INJ (2)</code>

- LL_ADC_AWD_CHANNEL_7_REG (2)
- LL_ADC_AWD_CHANNEL_7_INJ (2)
- LL_ADC_AWD_CHANNEL_7_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_8_REG (2)
- LL_ADC_AWD_CHANNEL_8_INJ (2)
- LL_ADC_AWD_CHANNEL_8_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_9_REG (2)
- LL_ADC_AWD_CHANNEL_9_INJ (2)
- LL_ADC_AWD_CHANNEL_9_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_10_REG (2)
- LL_ADC_AWD_CHANNEL_10_INJ (2)
- LL_ADC_AWD_CHANNEL_10_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_11_REG (2)
- LL_ADC_AWD_CHANNEL_11_INJ (2)
- LL_ADC_AWD_CHANNEL_11_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_12_REG (2)
- LL_ADC_AWD_CHANNEL_12_INJ (2)
- LL_ADC_AWD_CHANNEL_12_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_13_REG (3)
- LL_ADC_AWD_CHANNEL_13_INJ (3)
- LL_ADC_AWD_CHANNEL_13_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_14_REG (3)
- LL_ADC_AWD_CHANNEL_14_INJ (3)
- LL_ADC_AWD_CHANNEL_14_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_15_REG (3)
- LL_ADC_AWD_CHANNEL_15_INJ (3)
- LL_ADC_AWD_CHANNEL_15_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_16_REG (3)
- LL_ADC_AWD_CHANNEL_16_INJ (3)
- LL_ADC_AWD_CHANNEL_16_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_17_REG (3)
- LL_ADC_AWD_CHANNEL_17_INJ (3)
- LL_ADC_AWD_CHANNEL_17_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_18_REG (3)
- LL_ADC_AWD_CHANNEL_18_INJ (3)
- LL_ADC_AWD_CHANNEL_18_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_19_REG (3)
- LL_ADC_AWD_CHANNEL_19_INJ (3)
- LL_ADC_AWD_CHANNEL_19_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_20_REG (3)
- LL_ADC_AWD_CHANNEL_20_INJ (3)
- LL_ADC_AWD_CHANNEL_20_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_21_REG (3)
- LL_ADC_AWD_CHANNEL_21_INJ (3)
- LL_ADC_AWD_CHANNEL_21_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_22_REG (1)
- LL_ADC_AWD_CHANNEL_22_INJ (1)
- LL_ADC_AWD_CHANNEL_22_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_23_REG (1)
- LL_ADC_AWD_CHANNEL_23_INJ (1)
- LL_ADC_AWD_CHANNEL_23_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_24_REG (1)

- LL_ADC_AWD_CHANNEL_24_INJ (1)
- LL_ADC_AWD_CHANNEL_24_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_25_REG (1)
- LL_ADC_AWD_CHANNEL_25_INJ (1)
- LL_ADC_AWD_CHANNEL_25_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_26_REG (3)
- LL_ADC_AWD_CHANNEL_26_INJ (3)
- LL_ADC_AWD_CHANNEL_26_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_27_REG (3)(4)
- LL_ADC_AWD_CHANNEL_27_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_27_REG_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_28_REG (3)(4)
- LL_ADC_AWD_CHANNEL_28_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_28_REG_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_29_REG (3)(4)
- LL_ADC_AWD_CHANNEL_29_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_29_REG_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_30_REG (3)(4)
- LL_ADC_AWD_CHANNEL_30_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_30_REG_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_31_REG (3)(4)
- LL_ADC_AWD_CHANNEL_31_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_31_REG_INJ (3)(4)
- (2) On STM32L1, for devices with feature 'channels banks' available: Channel different in bank A and bank B.
- (3) On STM32L1, for devices with feature 'channels banks' available: Channel common to both bank A and bank B.
- (4) On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
- Usage of the returned channel number: To reinject this channel into another function LL_ADC_xxx: the returned channel number is only partly formatted on definition of literals LL_ADC_CHANNEL_x. Therefore, it has to be compared with parts of literals LL_ADC_CHANNEL_x or using helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB(). Then the selected literal LL_ADC_CHANNEL_x can be used as parameter for another function. To get the channel number in decimal format: process the returned value with the helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB(). Applicable only when the analog watchdog is set to monitor one channel.
- On this STM32 serie, there is only 1 kind of analog watchdog instance: AWD standard (instance AWD1): channels monitored: can monitor 1 channel or all channels.groups monitored: ADC groups regular and-or injected.resolution: resolution is not limited (corresponds to ADC resolution configured).

Notes

- CR1 AWD1CH LL_ADC_GetAnalogWDMonitChannels
- CR1 AWD1SGL LL_ADC_GetAnalogWDMonitChannels
- CR1 AWD1EN LL_ADC_GetAnalogWDMonitChannels

Reference Manual to
LL API cross
reference:

LL_ADC_SetAnalogWDThresholds

Function name	<code>_STATIC_INLINE void LL_ADC_SetAnalogWDThresholds(ADC_TypeDef * ADCx, uint32_t AWDThresholdsHighLow, uint32_t AWDThresholdValue)</code>
Function description	Set ADC analog watchdog threshold value of threshold high or low.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • AWDThresholdsHighLow: This parameter can be one of the following values: <ul style="list-style-type: none"> – <code>LL_ADC_AWD_THRESHOLD_HIGH</code> – <code>LL_ADC_AWD_THRESHOLD_LOW</code> • AWDThresholdValue: Value between Min_Data=0x000 and Max_Data=0xFFFF
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • In case of ADC resolution different of 12 bits, analog watchdog thresholds data require a specific shift. Use helper macro <code>_LL_ADC_ANALOGWD_SET_THRESHOLD_RESOLUTION()</code>. • On this STM32 serie, there is only 1 kind of analog watchdog instance: AWD standard (instance AWD1): channels monitored: can monitor 1 channel or all channels.groups monitored: ADC groups regular and-or injected.resolution: resolution is not limited (corresponds to ADC resolution configured).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • HTR HT LL_ADC_SetAnalogWDThresholds • LTR LT LL_ADC_SetAnalogWDThresholds

LL_ADC_GetAnalogWDThresholds

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_GetAnalogWDThresholds(ADC_TypeDef * ADCx, uint32_t AWDThresholdsHighLow)</code>
Function description	Get ADC analog watchdog threshold value of threshold high or threshold low.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • AWDThresholdsHighLow: This parameter can be one of the following values: <ul style="list-style-type: none"> – <code>LL_ADC_AWD_THRESHOLD_HIGH</code> – <code>LL_ADC_AWD_THRESHOLD_LOW</code>
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x000 and Max_Data=0xFFFF
Notes	<ul style="list-style-type: none"> • In case of ADC resolution different of 12 bits, analog watchdog thresholds data require a specific shift. Use helper macro <code>_LL_ADC_ANALOGWD_GET_THRESHOLD_RESOLUTION()</code>.
Reference Manual to LL API cross	<ul style="list-style-type: none"> • HTR HT LL_ADC_GetAnalogWDThresholds • LTR LT LL_ADC_GetAnalogWDThresholds

reference:

LL_ADC_Enable

Function name	<code>_STATIC_INLINE void LL_ADC_Enable (ADC_TypeDef * ADCx)</code>
Function description	Enable the selected ADC instance.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • On this STM32 serie, after ADC enable, a delay for ADC internal analog stabilization is required before performing a ADC conversion start. Refer to device datasheet, parameter tSTAB. • Due to the latency introduced by the synchronization between two clock domains (ADC clock source asynchronous), some hardware constraints must be respected: ADC must be enabled (LL_ADC_Enable()) only when ADC is not ready to convert.ADC must be disabled (LL_ADC_Disable()) only when ADC is ready to convert. Status of ADC ready to convert can be checked using function LL_ADC_IsActiveFlag_ADRDY().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ADON LL_ADC_Enable

LL_ADC_Disable

Function name	<code>_STATIC_INLINE void LL_ADC_Disable (ADC_TypeDef * ADCx)</code>
Function description	Disable the selected ADC instance.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Due to the latency introduced by the synchronization between two clock domains (ADC clock source asynchronous), some hardware constraints must be respected: ADC must be enabled (LL_ADC_Enable()) only when ADC is not ready to convert.ADC must be disabled (LL_ADC_Disable()) only when ADC is ready to convert. Status of ADC ready to convert can be checked using function LL_ADC_IsActiveFlag_ADRDY().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ADON LL_ADC_Disable

LL_ADC_IsEnabled

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_IsEnabled (ADC_TypeDef * ADCx)</code>
---------------	---

Function description	Get the selected ADC instance enable state.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> 0: ADC is disabled, 1: ADC is enabled.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ADON LL_ADC_IsEnabled

LL_ADC_REG_StartConversionSWStart

Function name	<code>__STATIC_INLINE void LL_ADC_REG_StartConversionSWStart (ADC_TypeDef * ADCx)</code>
Function description	Start ADC group regular conversion.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> On this STM32 serie, this function is relevant only for internal trigger (SW start), not for external trigger: If ADC trigger has been set to software start, ADC conversion starts immediately. If ADC trigger has been set to external trigger, ADC conversion start must be performed using function <code>LL_ADC_REG_StartConversionExtTrig()</code>. (if external trigger edge would have been set during ADC other settings, ADC conversion would start at trigger event as soon as ADC is enabled).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 SWSTART LL_ADC_REG_StartConversionSWStart

LL_ADC_REG_StartConversionExtTrig

Function name	<code>__STATIC_INLINE void LL_ADC_REG_StartConversionExtTrig (ADC_TypeDef * ADCx, uint32_t ExternalTriggerEdge)</code>
Function description	Start ADC group regular conversion from external trigger.
Parameters	<ul style="list-style-type: none"> ExternalTriggerEdge: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_ADC_REG_TRIG_EXT_RISING</code> - <code>LL_ADC_REG_TRIG_EXT_FALLING</code> - <code>LL_ADC_REG_TRIG_EXT_RISINGFALLING</code> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> ADC conversion will start at next trigger event (on the selected trigger edge) following the ADC start conversion command. On this STM32 serie, this function is relevant for ADC conversion start from external trigger. If internal trigger (SW start) is needed, perform ADC conversion start using function <code>LL_ADC_REG_StartConversionSWStart()</code>.

Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 EXTEN LL_ADC_REG_StartConversionExtTrig
Function name	<code>_STATIC_INLINE void LL_ADC_REG_StopConversionExtTrig (ADC_TypeDef * ADCx)</code>
Function description	Stop ADC group regular conversion from external trigger.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • No more ADC conversion will start at next trigger event following the ADC stop conversion command. If a conversion is on-going, it will be completed. • On this STM32 serie, there is no specific command to stop a conversion on-going or to stop ADC converting in continuous mode. These actions can be performed using function <code>LL_ADC_Disable()</code>.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 EXTEN LL_ADC_REG_StopConversionExtTrig

LL_ADC_REG_ReadConversionData32

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_REG_ReadConversionData32 (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion data, range fit for all ADC configurations: all ADC resolutions and all oversampling increased data width (for devices with feature oversampling).
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00000000 and Max_Data=0xFFFFFFFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DR RDATA LL_ADC_REG_ReadConversionData32

LL_ADC_REG_ReadConversionData12

Function name	<code>_STATIC_INLINE uint16_t LL_ADC_REG_ReadConversionData12 (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion data, range fit for ADC resolution 12 bits.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x000 and Max_Data=0xFFFF
Notes	<ul style="list-style-type: none"> • For devices with feature oversampling: Oversampling can increase data width, function for extended range may be

needed: LL_ADC_REG_ReadConversionData32.

Reference Manual to
LL API cross
reference:

- DR RDATA LL_ADC_REG_ReadConversionData12

LL_ADC_REG_ReadConversionData10

Function name	<code>__STATIC_INLINE uint16_t LL_ADC_REG_ReadConversionData10 (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion data, range fit for ADC resolution 10 bits.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x000 and Max_Data=0x3FF
Notes	<ul style="list-style-type: none"> For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_REG_ReadConversionData32.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DR RDATA LL_ADC_REG_ReadConversionData10

LL_ADC_REG_ReadConversionData8

Function name	<code>__STATIC_INLINE uint8_t LL_ADC_REG_ReadConversionData8 (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion data, range fit for ADC resolution 8 bits.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x00 and Max_Data=0xFF
Notes	<ul style="list-style-type: none"> For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_REG_ReadConversionData32.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DR RDATA LL_ADC_REG_ReadConversionData8

LL_ADC_REG_ReadConversionData6

Function name	<code>__STATIC_INLINE uint8_t LL_ADC_REG_ReadConversionData6 (ADC_TypeDef * ADCx)</code>
Function description	Get ADC group regular conversion data, range fit for ADC resolution 6 bits.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x00 and Max_Data=0x3F
Notes	<ul style="list-style-type: none"> For devices with feature oversampling: Oversampling can increase data width, function for extended range may be

Reference Manual to
LL API cross
reference:

needed: LL_ADC_REG_ReadConversionData32.

- DR RDATA LL_ADC_REG_ReadConversionData6

LL_ADC_INJ_StartConversionSWStart

Function name	<u>__STATIC_INLINE void LL_ADC_INJ_StartConversionSWStart(ADC_TypeDef * ADCx)</u>
Function description	Start ADC group injected conversion.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • On this STM32 serie, this function is relevant only for internal trigger (SW start), not for external trigger: If ADC trigger has been set to software start, ADC conversion starts immediately. If ADC trigger has been set to external trigger, ADC conversion start must be performed using function LL_ADC_INJ_StartConversionExtTrig(). (if external trigger edge would have been set during ADC other settings, ADC conversion would start at trigger event as soon as ADC is enabled).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 JSWSTART LL_ADC_INJ_StartConversionSWStart

LL_ADC_INJ_StartConversionExtTrig

Function name	<u>__STATIC_INLINE void LL_ADC_INJ_StartConversionExtTrig(ADC_TypeDef * ADCx, uint32_t ExternalTriggerEdge)</u>
Function description	Start ADC group injected conversion from external trigger.
Parameters	<ul style="list-style-type: none"> • ExternalTriggerEdge: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_INJ_TRIG_EXT_RISING - LL_ADC_INJ_TRIG_EXT_FALLING - LL_ADC_INJ_TRIG_EXT_RISINGFALLING • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • ADC conversion will start at next trigger event (on the selected trigger edge) following the ADC start conversion command. • On this STM32 serie, this function is relevant for ADC conversion start from external trigger. If internal trigger (SW start) is needed, perform ADC conversion start using function LL_ADC_INJ_StartConversionSWStart().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 JEXTEN LL_ADC_INJ_StartConversionExtTrig

LL_ADC_INJ_StopConversionExtTrig

Function name	<code>__STATIC_INLINE void LL_ADC_INJ_StopConversionExtTrig(ADC_TypeDef * ADCx)</code>
Function description	Stop ADC group injected conversion from external trigger.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • No more ADC conversion will start at next trigger event following the ADC stop conversion command. If a conversion is on-going, it will be completed. • On this STM32 serie, there is no specific command to stop a conversion on-going or to stop ADC converting in continuous mode. These actions can be performed using function <code>LL_ADC_Disable()</code>.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 JEXTEN LL_ADC_INJ_StopConversionExtTrig

LL_ADC_INJ_ReadConversionData32

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_INJ_ReadConversionData32 (ADC_TypeDef * ADCx, uint32_t Rank)</code>
Function description	Get ADC group regular conversion data, range fit for all ADC configurations: all ADC resolutions and all oversampling increased data width (for devices with feature oversampling).
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Rank: This parameter can be one of the following values: <ul style="list-style-type: none"> – <code>LL_ADC_INJ_RANK_1</code> – <code>LL_ADC_INJ_RANK_2</code> – <code>LL_ADC_INJ_RANK_3</code> – <code>LL_ADC_INJ_RANK_4</code>
Return values	<ul style="list-style-type: none"> • Value: between <code>Min_Data=0x00000000</code> and <code>Max_Data=0xFFFFFFFF</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • JDR1 JDATA <code>LL_ADC_INJ_ReadConversionData32</code> • JDR2 JDATA <code>LL_ADC_INJ_ReadConversionData32</code> • JDR3 JDATA <code>LL_ADC_INJ_ReadConversionData32</code> • JDR4 JDATA <code>LL_ADC_INJ_ReadConversionData32</code>

LL_ADC_INJ_ReadConversionData12

Function name	<code>__STATIC_INLINE uint16_t LL_ADC_INJ_ReadConversionData12 (ADC_TypeDef * ADCx, uint32_t Rank)</code>
Function description	Get ADC group injected conversion data, range fit for ADC resolution 12 bits.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Rank: This parameter can be one of the following values: <ul style="list-style-type: none"> – <code>LL_ADC_INJ_RANK_1</code>

	<ul style="list-style-type: none"> - LL_ADC_INJ_RANK_2 - LL_ADC_INJ_RANK_3 - LL_ADC_INJ_RANK_4
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x000 and Max_Data=0xFFFF
Notes	<ul style="list-style-type: none"> • For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_INJ_ReadConversionData32.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • JDR1 JDATA LL_ADC_INJ_ReadConversionData12 • JDR2 JDATA LL_ADC_INJ_ReadConversionData12 • JDR3 JDATA LL_ADC_INJ_ReadConversionData12 • JDR4 JDATA LL_ADC_INJ_ReadConversionData12

LL_ADC_INJ_ReadConversionData10

Function name	<code>__STATIC_INLINE uint16_t LL_ADC_INJ_ReadConversionData10 (ADC_TypeDef * ADCx, uint32_t Rank)</code>
Function description	Get ADC group injected conversion data, range fit for ADC resolution 10 bits.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Rank: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_INJ_RANK_1 - LL_ADC_INJ_RANK_2 - LL_ADC_INJ_RANK_3 - LL_ADC_INJ_RANK_4
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x000 and Max_Data=0x3FF
Notes	<ul style="list-style-type: none"> • For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_INJ_ReadConversionData32.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • JDR1 JDATA LL_ADC_INJ_ReadConversionData10 • JDR2 JDATA LL_ADC_INJ_ReadConversionData10 • JDR3 JDATA LL_ADC_INJ_ReadConversionData10 • JDR4 JDATA LL_ADC_INJ_ReadConversionData10

LL_ADC_INJ_ReadConversionData8

Function name	<code>__STATIC_INLINE uint8_t LL_ADC_INJ_ReadConversionData8 (ADC_TypeDef * ADCx, uint32_t Rank)</code>
Function description	Get ADC group injected conversion data, range fit for ADC resolution 8 bits.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • Rank: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_ADC_INJ_RANK_1 - LL_ADC_INJ_RANK_2 - LL_ADC_INJ_RANK_3 - LL_ADC_INJ_RANK_4

Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x00 and Max_Data=0xFF
Notes	<ul style="list-style-type: none"> For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_INJ_ReadConversionData32.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> JDR1 JDATA LL_ADC_INJ_ReadConversionData8 JDR2 JDATA LL_ADC_INJ_ReadConversionData8 JDR3 JDATA LL_ADC_INJ_ReadConversionData8 JDR4 JDATA LL_ADC_INJ_ReadConversionData8

LL_ADC_INJ_ReadConversionData6

Function name	<code>__STATIC_INLINE uint8_t LL_ADC_INJ_ReadConversionData6 (ADC_TypeDef * ADCx, uint32_t Rank)</code>
Function description	Get ADC group injected conversion data, range fit for ADC resolution 6 bits.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance Rank: This parameter can be one of the following values: <ul style="list-style-type: none"> LL_ADC_INJ_RANK_1 LL_ADC_INJ_RANK_2 LL_ADC_INJ_RANK_3 LL_ADC_INJ_RANK_4
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x00 and Max_Data=0x3F
Notes	<ul style="list-style-type: none"> For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_INJ_ReadConversionData32.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> JDR1 JDATA LL_ADC_INJ_ReadConversionData6 JDR2 JDATA LL_ADC_INJ_ReadConversionData6 JDR3 JDATA LL_ADC_INJ_ReadConversionData6 JDR4 JDATA LL_ADC_INJ_ReadConversionData6

LL_ADC_IsActiveFlag_ADRDY

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_ADRDY (ADC_TypeDef * ADCx)</code>
Function description	Get flag ADC ready.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).

Reference Manual to
LL API cross
reference:
SR ADONS LL_ADC_IsActiveFlag_ADRDY

LL_ADC_IsActiveFlag_EOCS

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_EOCS (ADC_TypeDef * ADCx)</code>
Function description	Get flag ADC group regular end of unitary conversion or end of

sequence conversions, depending on ADC configuration.

Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR EOC LL_ADC_IsActiveFlag_EOCS

LL_ADC_IsActiveFlag_OVR

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_OVR(ADC_TypeDef * ADCx)</code>
Function description	Get flag ADC group regular overrun.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR OVR LL_ADC_IsActiveFlag_OVR

LL_ADC_IsActiveFlag_JEOS

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_JEOS(ADC_TypeDef * ADCx)</code>
Function description	Get flag ADC group injected end of sequence conversions.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR JEOC LL_ADC_IsActiveFlag_JEOS

LL_ADC_IsActiveFlag_AWD1

Function name	<code>__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_AWD1(ADC_TypeDef * ADCx)</code>
Function description	Get flag ADC analog watchdog 1 flag.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR AWD LL_ADC_IsActiveFlag_AWD1

LL_ADC_ClearFlag_EOCS

Function name	<code>__STATIC_INLINE void LL_ADC_ClearFlag_EOCS</code>
---------------	---

(ADC_TypeDef * ADCx)

Function description	Clear flag ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR EOC LL_ADC_ClearFlag_EOCS

LL_ADC_ClearFlag_OVR

Function name	_STATIC_INLINE void LL_ADC_ClearFlag_OVR (ADC_TypeDef * ADCx)
Function description	Clear flag ADC group regular overrun.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR OVR LL_ADC_ClearFlag_OVR

LL_ADC_ClearFlag_JEOS

Function name	_STATIC_INLINE void LL_ADC_ClearFlag_JEOS (ADC_TypeDef * ADCx)
Function description	Clear flag ADC group injected end of sequence conversions.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR JEOC LL_ADC_ClearFlag_JEOS

LL_ADC_ClearFlag_AWD1

Function name	_STATIC_INLINE void LL_ADC_ClearFlag_AWD1 (ADC_TypeDef * ADCx)
Function description	Clear flag ADC analog watchdog 1.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR AWD LL_ADC_ClearFlag_AWD1

LL_ADC_EnableIT_EOCS

Function name	<code>__STATIC_INLINE void LL_ADC_EnableIT_EOCS(ADC_TypeDef * ADCx)</code>
Function description	Enable interruption ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • To configure flag of end of conversion, use function <code>LL_ADC_REG_SetFlagEndOfConversion()</code>.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 EOcie LL_ADC_EnableIT_EOCS

LL_ADC_EnableIT_OVR

Function name	<code>__STATIC_INLINE void LL_ADC_EnableIT_OVR(ADC_TypeDef * ADCx)</code>
Function description	Enable ADC group regular interruption overrun.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 OVRIE LL_ADC_EnableIT_OVR

LL_ADC_EnableIT_JEOS

Function name	<code>__STATIC_INLINE void LL_ADC_EnableIT_JEOS(ADC_TypeDef * ADCx)</code>
Function description	Enable interruption ADC group injected end of sequence conversions.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 JEOCIE LL_ADC_EnableIT_JEOS

LL_ADC_EnableIT_AWD1

Function name	<code>__STATIC_INLINE void LL_ADC_EnableIT_AWD1(ADC_TypeDef * ADCx)</code>
Function description	Enable interruption ADC analog watchdog 1.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross	<ul style="list-style-type: none"> • CR1 AWDIE LL_ADC_EnableIT_AWD1

reference:

LL_ADC_DisableIT_EOCS

Function name	<code>_STATIC_INLINE void LL_ADC_DisableIT_EOCS(ADC_TypeDef * ADCx)</code>
Function description	Disable interruption ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • To configure flag of end of conversion, use function <code>LL_ADC_REG_SetFlagEndOfConversion()</code>.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 EOCIE LL_ADC_DisableIT_EOCS

LL_ADC_DisableIT_OVR

Function name	<code>_STATIC_INLINE void LL_ADC_DisableIT_OVR(ADC_TypeDef * ADCx)</code>
Function description	Disable interruption ADC group regular overrun.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 OVRIE LL_ADC_DisableIT_OVR

LL_ADC_DisableIT_JEOS

Function name	<code>_STATIC_INLINE void LL_ADC_DisableIT_JEOS(ADC_TypeDef * ADCx)</code>
Function description	Disable interruption ADC group injected end of sequence conversions.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 JEOCIE LL_ADC_EnableIT_JEOS

LL_ADC_DisableIT_AWD1

Function name	<code>_STATIC_INLINE void LL_ADC_DisableIT_AWD1(ADC_TypeDef * ADCx)</code>
Function description	Disable interruption ADC analog watchdog 1.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance

Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 AWDIE LL_ADC_EnableIT_AWD1

LL_ADC_IsEnabledIT_EOCS

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_EOCS(ADC_TypeDef * ADCx)</code>
Function description	Get state of interruption ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> To configure flag of end of conversion, use function <code>LL_ADC_REG_SetFlagEndOfConversion()</code>. (0: interrupt disabled, 1: interrupt enabled)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 EOCIE LL_ADC_IsEnabledIT_EOCS

LL_ADC_IsEnabledIT_OVR

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_OVR(ADC_TypeDef * ADCx)</code>
Function description	Get state of interruption ADC group regular overrun (0: interrupt disabled, 1: interrupt enabled).
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 OVRIE LL_ADC_IsEnabledIT_OVR

LL_ADC_IsEnabledIT_JEOS

Function name	<code>_STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_JEOS(ADC_TypeDef * ADCx)</code>
Function description	Get state of interruption ADC group injected end of sequence conversions (0: interrupt disabled, 1: interrupt enabled).
Parameters	<ul style="list-style-type: none"> ADCx: ADC instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 JEOCIE LL_ADC_EnableIT_JEOS

LL_ADC_IsEnabledIT_AWD1

Function name **`_STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_AWD1(ADC_TypeDef * ADCx)`**

Function description Get state of interruption ADC analog watchdog 1 (0: interrupt disabled, 1: interrupt enabled).

Parameters • **ADCx:** ADC instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
CR1 AWDIE LL_ADC_EnableIT_AWD1

LL_ADC_CommonDeInit

Function name **`ErrorStatus LL_ADC_CommonDeInit (ADC_Common_TypeDef * ADCxy_COMMON)`**

Function description De-initialize registers of all ADC instances belonging to the same ADC common instance to their default reset values.

Parameters • **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro `_LL_ADC_COMMON_INSTANCE()`)

Return values • **An:** ErrorStatus enumeration value:
– SUCCESS: ADC common registers are de-initialized
– ERROR: not applicable

LL_ADC_CommonInit

Function name **`ErrorStatus LL_ADC_CommonInit (ADC_Common_TypeDef * ADCxy_COMMON, LL_ADC_CommonInitTypeDef * ADC_CommonInitStruct)`**

Function description Initialize some features of ADC common parameters (all ADC instances belonging to the same ADC common instance) and multimode (for devices with several ADC instances available).

Parameters • **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro `_LL_ADC_COMMON_INSTANCE()`)
• **ADC_CommonInitStruct:** Pointer to a `LL_ADC_CommonInitStruct` structure

Return values • **An:** ErrorStatus enumeration value:
– SUCCESS: ADC common registers are initialized
– ERROR: ADC common registers are not initialized

Notes • The setting of ADC common parameters is conditioned to ADC instances state: All ADC instances belonging to the same ADC common instance must be disabled.

LL_ADC_CommonStructInit

Function name **`void LL_ADC_CommonStructInit`**

(LL_ADC_CommonInitTypeDef * ADC_CommonInitStruct)

Function description	Set each LL_ADC_CommonInitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • ADC_CommonInitStruct: Pointer to a LL_ADC_CommonInitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> • None:

LL_ADC_Delnit

Function name	ErrorStatus LL_ADC_Delnit (ADC_TypeDef * ADCx)
Function description	De-initialize registers of the selected ADC instance to their default reset values.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: ADC registers are de-initialized – ERROR: ADC registers are not de-initialized
Notes	<ul style="list-style-type: none"> • To reset all ADC instances quickly (perform a hard reset), use function LL_ADC_CommonDelnit().

LL_ADC_Init

Function name	ErrorStatus LL_ADC_Init (ADC_TypeDef * ADCx, LL_ADC_InitTypeDef * ADC_InitStruct)
Function description	Initialize some features of ADC instance.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • ADC_InitStruct: Pointer to a LL_ADC_REG_InitTypeDef structure
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: ADC registers are initialized – ERROR: ADC registers are not initialized
Notes	<ul style="list-style-type: none"> • These parameters have an impact on ADC scope: ADC instance. Affects both group regular and group injected (availability of ADC group injected depends on STM32 families). Refer to corresponding unitary functions into Configuration of ADC hierarchical scope: ADC instance . • The setting of these parameters by function LL_ADC_Init() is conditioned to ADC state: ADC instance must be disabled. This condition is applied to all ADC features, for efficiency and compatibility over all STM32 families. However, the different features can be set under different ADC state conditions (setting possible with ADC enabled without conversion on going, ADC enabled with conversion on going, ...) Each feature can be updated afterwards with a unitary function and potentially with ADC in a different state than disabled, refer to description of each function for setting conditioned to ADC state. • After using this function, some other features must be configured using LL unitary functions. The minimum configuration remaining to be done is: Set ADC group regular

or group injected sequencer: map channel on the selected sequencer rank. Refer to function
LL_ADC_REG_SetSequencerRanks().Set ADC channel sampling time Refer to function
LL_ADC_SetChannelSamplingTime();

LL_ADC_StructInit

Function name	void LL_ADC_StructInit (LL_ADC_InitTypeDef * ADC_InitStruct)
Function description	Set each LL_ADC_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • ADC_InitStruct: Pointer to a LL_ADC_InitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> • None:

LL_ADC_REG_Init

Function name	ErrorStatus LL_ADC_REG_Init (ADC_TypeDef * ADCx, LL_ADC_InitTypeDef * ADC_REG_InitStruct)
Function description	Initialize some features of ADC group regular.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • ADC_REG_InitStruct: Pointer to a LL_ADC_REG_InitTypeDef structure
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: ADC registers are initialized – ERROR: ADC registers are not initialized
Notes	<ul style="list-style-type: none"> • These parameters have an impact on ADC scope: ADC group regular. Refer to corresponding unitary functions into Configuration of ADC hierarchical scope: group regular (functions with prefix "REG"). • The setting of these parameters by function LL_ADC_Init() is conditioned to ADC state: ADC instance must be disabled. This condition is applied to all ADC features, for efficiency and compatibility over all STM32 families. However, the different features can be set under different ADC state conditions (setting possible with ADC enabled without conversion on going, ADC enabled with conversion on going, ...) Each feature can be updated afterwards with a unitary function and potentially with ADC in a different state than disabled, refer to description of each function for setting conditioned to ADC state. • After using this function, other features must be configured using LL unitary functions. The minimum configuration remaining to be done is: Set ADC group regular or group injected sequencer: map channel on the selected sequencer rank. Refer to function LL_ADC_REG_SetSequencerRanks().Set ADC channel sampling time Refer to function LL_ADC_SetChannelSamplingTime();

LL_ADC_REG_StructInit

Function name	<code>void LL_ADC_REG_StructInit (LL_ADC_REG_InitTypeDef * ADC_REG_InitStruct)</code>
Function description	Set each LL_ADC_REG_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • ADC_REG_InitStruct: Pointer to a LL_ADC_REG_InitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> • None:

LL_ADC_INJ_Init

Function name	<code>ErrorStatus LL_ADC_INJ_Init (ADC_TypeDef * ADCx, LL_ADC_INJ_InitTypeDef * ADC_INJ_InitStruct)</code>
Function description	Initialize some features of ADC group injected.
Parameters	<ul style="list-style-type: none"> • ADCx: ADC instance • ADC_INJ_InitStruct: Pointer to a LL_ADC_INJ_InitTypeDef structure
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: ADC registers are initialized – ERROR: ADC registers are not initialized
Notes	<ul style="list-style-type: none"> • These parameters have an impact on ADC scope: ADC group injected. Refer to corresponding unitary functions into Configuration of ADC hierarchical scope: group regular (functions with prefix "INJ"). • The setting of these parameters by function LL_ADC_Init() is conditioned to ADC state: ADC instance must be disabled. This condition is applied to all ADC features, for efficiency and compatibility over all STM32 families. However, the different features can be set under different ADC state conditions (setting possible with ADC enabled without conversion on going, ADC enabled with conversion on going, ...) Each feature can be updated afterwards with a unitary function and potentially with ADC in a different state than disabled, refer to description of each function for setting conditioned to ADC state. • After using this function, other features must be configured using LL unitary functions. The minimum configuration remaining to be done is: Set ADC group injected sequencer: map channel on the selected sequencer rank. Refer to function LL_ADC_INJ_SetSequencerRanks(). Set ADC channel sampling time Refer to function LL_ADC_SetChannelSamplingTime();

LL_ADC_INJ_StructInit

Function name	<code>void LL_ADC_INJ_StructInit (LL_ADC_INJ_InitTypeDef * ADC_INJ_InitStruct)</code>
Function description	Set each LL_ADC_INJ_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • ADC_INJ_InitStruct: Pointer to a LL_ADC_INJ_InitTypeDef

structure whose fields will be set to default values.

Return values

- **None:**

48.3 ADC Firmware driver defines

48.3.1 ADC

Analog watchdog - Monitored channels

<code>LL_ADC_AWD_DISABLE</code>	ADC analog watchdog monitoring disabled
<code>LL_ADC_AWD_ALL_CHANNELS_REG</code>	ADC analog watchdog monitoring of all channels, converted by group regular only
<code>LL_ADC_AWD_ALL_CHANNELS_INJ</code>	ADC analog watchdog monitoring of all channels, converted by group injected only
<code>LL_ADC_AWD_ALL_CHANNELS_REG_INJ</code>	ADC analog watchdog monitoring of all channels, converted by either group regular or injected
<code>LL_ADC_AWD_CHANNEL_0_REG</code>	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN0, converted by group regular only
<code>LL_ADC_AWD_CHANNEL_0_INJ</code>	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN0, converted by group injected only
<code>LL_ADC_AWD_CHANNEL_0_REG_INJ</code>	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN0, converted by either group regular or injected
<code>LL_ADC_AWD_CHANNEL_1_REG</code>	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN1, converted by group regular only
<code>LL_ADC_AWD_CHANNEL_1_INJ</code>	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN1, converted by group injected only
<code>LL_ADC_AWD_CHANNEL_1_REG_INJ</code>	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN1, converted by either group regular or injected
<code>LL_ADC_AWD_CHANNEL_2_REG</code>	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN2,

LL_ADC_AWD_CHANNEL_2_INJ	converted by group regular only ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN2, converted by group injected only
LL_ADC_AWD_CHANNEL_2_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN2, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_3_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN3, converted by group regular only
LL_ADC_AWD_CHANNEL_3_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN3, converted by group injected only
LL_ADC_AWD_CHANNEL_3_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN3, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_4_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN4, converted by group regular only
LL_ADC_AWD_CHANNEL_4_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN4, converted by group injected only
LL_ADC_AWD_CHANNEL_4_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN4, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_5_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN5, converted by group regular only
LL_ADC_AWD_CHANNEL_5_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN5, converted by group injected only
LL_ADC_AWD_CHANNEL_5_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN5, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_6_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN6, converted by group regular only
LL_ADC_AWD_CHANNEL_6_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN6, converted by group injected only
LL_ADC_AWD_CHANNEL_6_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN6, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_7_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN7, converted by group regular only
LL_ADC_AWD_CHANNEL_7_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN7, converted by group injected only
LL_ADC_AWD_CHANNEL_7_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN7, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_8_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN8, converted by group regular only
LL_ADC_AWD_CHANNEL_8_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN8, converted by group injected only
LL_ADC_AWD_CHANNEL_8_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN8, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_9_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN9, converted by group regular only
LL_ADC_AWD_CHANNEL_9_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN9, converted by group injected only
LL_ADC_AWD_CHANNEL_9_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN9, converted by either group regular or injected

	converted by either group regular or injected
LL_ADC_AWD_CHANNEL_10_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN10, converted by group regular only
LL_ADC_AWD_CHANNEL_10_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN10, converted by group injected only
LL_ADC_AWD_CHANNEL_10_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN10, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_11_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN11, converted by group regular only
LL_ADC_AWD_CHANNEL_11_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN11, converted by group injected only
LL_ADC_AWD_CHANNEL_11_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN11, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_12_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN12, converted by group regular only
LL_ADC_AWD_CHANNEL_12_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN12, converted by group injected only
LL_ADC_AWD_CHANNEL_12_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN12, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_13_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN13, converted by group regular only
LL_ADC_AWD_CHANNEL_13_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN13, converted by group injected only

LL_ADC_AWD_CHANNEL_13_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN13, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_14_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN14, converted by group regular only
LL_ADC_AWD_CHANNEL_14_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN14, converted by group injected only
LL_ADC_AWD_CHANNEL_14_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN14, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_15_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN15, converted by group regular only
LL_ADC_AWD_CHANNEL_15_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN15, converted by group injected only
LL_ADC_AWD_CHANNEL_15_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN15, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_16_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN16, converted by group regular only
LL_ADC_AWD_CHANNEL_16_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN16, converted by group injected only
LL_ADC_AWD_CHANNEL_16_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN16, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_17_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN17, converted by group regular only
LL_ADC_AWD_CHANNEL_17_INJ	ADC analog watchdog monitoring of ADC external channel (channel

	connected to GPIO pin) ADCx_IN17, converted by group injected only
LL_ADC_AWD_CHANNEL_17_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN17, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_18_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN18, converted by group regular only
LL_ADC_AWD_CHANNEL_18_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN18, converted by group injected only
LL_ADC_AWD_CHANNEL_18_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN18, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_19_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN19, converted by group regular only
LL_ADC_AWD_CHANNEL_19_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN19, converted by group injected only
LL_ADC_AWD_CHANNEL_19_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN19, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_20_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN20, converted by group regular only
LL_ADC_AWD_CHANNEL_20_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN20, converted by group injected only
LL_ADC_AWD_CHANNEL_20_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN20, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_21_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN21,

LL_ADC_AWD_CHANNEL_21_INJ	converted by group regular only ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN21, converted by group injected only
LL_ADC_AWD_CHANNEL_21_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN21, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_22_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN22, converted by group regular only
LL_ADC_AWD_CHANNEL_22_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN22, converted by group injected only
LL_ADC_AWD_CHANNEL_22_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN22, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_23_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN23, converted by group regular only
LL_ADC_AWD_CHANNEL_23_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN23, converted by group injected only
LL_ADC_AWD_CHANNEL_23_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN23, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_24_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN24, converted by group regular only
LL_ADC_AWD_CHANNEL_24_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN24, converted by group injected only
LL_ADC_AWD_CHANNEL_24_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN24, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_25_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN25, converted by group regular only
LL_ADC_AWD_CHANNEL_25_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN25, converted by group injected only
LL_ADC_AWD_CHANNEL_25_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN25, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_26_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN26, converted by group regular only
LL_ADC_AWD_CHANNEL_26_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN26, converted by group injected only
LL_ADC_AWD_CHANNEL_26_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN26, converted by either group regular or injected
LL_ADC_AWD_CHANNEL_27_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN27, converted by group regular only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_AWD_CHANNEL_27_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN27, converted by group injected only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_AWD_CHANNEL_27_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN27, converted by either group regular or injected. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_AWD_CHANNEL_28_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN28, converted by group regular only. On

STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.

`LL_ADC_AWD_CHANNEL_28_INJ`

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN28, converted by group injected only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.

`LL_ADC_AWD_CHANNEL_28_REG_INJ`

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN28, converted by either group regular or injected. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.

`LL_ADC_AWD_CHANNEL_29_REG`

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN29, converted by group regular only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.

`LL_ADC_AWD_CHANNEL_29_INJ`

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN29, converted by group injected only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.

`LL_ADC_AWD_CHANNEL_29_REG_INJ`

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN29, converted by either group regular or injected. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.

`LL_ADC_AWD_CHANNEL_30_REG`

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN30, converted by group regular only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.

`LL_ADC_AWD_CHANNEL_30_INJ`

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN30, converted by group injected only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and

	Cat.5.
LL_ADC_AWD_CHANNEL_30_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN30, converted by either group regular or injected. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_AWD_CHANNEL_31_REG	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN31, converted by group regular only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_AWD_CHANNEL_31_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN31, converted by group injected only. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_AWD_CHANNEL_31_REG_INJ	ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN31, converted by either group regular or injected. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_AWD_CH_VREFINT_REG	ADC analog watchdog monitoring of ADC internal channel connected to VrefInt: Internal voltage reference, converted by group regular only. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VREFINT_INJ	ADC analog watchdog monitoring of ADC internal channel connected to VrefInt: Internal voltage reference, converted by group injected only. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VREFINT_REG_INJ	ADC analog watchdog monitoring of ADC internal channel connected to VrefInt: Internal voltage reference, converted by either group regular or injected. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_TEMPSENSOR_REG	ADC analog watchdog monitoring of ADC internal channel connected to Temperature sensor, converted by group regular only. Channel common to both

	bank A and bank B.
LL_ADC_AWD_CH_TEMPSENSOR_INJ	ADC analog watchdog monitoring of ADC internal channel connected to Temperature sensor, converted by group injected only. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_TEMPSENSOR_REG_INJ	ADC analog watchdog monitoring of ADC internal channel connected to Temperature sensor, converted by either group regular or injected. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VCOMP_REG	ADC analog watchdog monitoring of ADC internal channel connected to comparator COMP1 positive input via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VCOMP_INJ	ADC analog watchdog monitoring of ADC internal channel connected to comparator COMP1 positive input via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VCOMP_REG_INJ	ADC analog watchdog monitoring of ADC internal channel connected to comparator COMP1 positive input via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VOPAMP1_REG	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP1 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VOPAMP1_INJ	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP1 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VOPAMP1_REG_INJ	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP1 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VOPAMP2_REG	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP2 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VOPAMP2_INJ	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP2 output via ADC switch matrix. Channel common to both bank A and

	bank B.
LL_ADC_AWD_CH_VOPAMP2_REG_INJ	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP2 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VOPAMP3_REG	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP3 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VOPAMP3_INJ	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP3 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_AWD_CH_VOPAMP3_REG_INJ	ADC analog watchdog monitoring of ADC internal channel connected to OPAMP3 output via ADC switch matrix. Channel common to both bank A and bank B.

Analog watchdog - Analog watchdog number

LL_ADC_AWD1 ADC analog watchdog number 1

Analog watchdog - Thresholds

LL_ADC_AWD_THRESHOLD_HIGH ADC analog watchdog threshold high

LL_ADC_AWD_THRESHOLD_LOW ADC analog watchdog threshold low

ADC instance - Channel number

LL_ADC_CHANNEL_0 ADC external channel (channel connected to GPIO pin) ADCx_IN0 . Channel different in bank A and bank B.

LL_ADC_CHANNEL_1 ADC external channel (channel connected to GPIO pin) ADCx_IN1 . Channel different in bank A and bank B.

LL_ADC_CHANNEL_2 ADC external channel (channel connected to GPIO pin) ADCx_IN2 . Channel different in bank A and bank B.

LL_ADC_CHANNEL_3 ADC external channel (channel connected to GPIO pin) ADCx_IN3 . Channel different in bank A and bank B.

LL_ADC_CHANNEL_4 ADC external channel (channel connected to GPIO pin) ADCx_IN4 . Direct (fast) channel.

LL_ADC_CHANNEL_5 ADC external channel (channel connected to GPIO pin) ADCx_IN5 . Direct (fast) channel.

LL_ADC_CHANNEL_6 ADC external channel (channel connected to GPIO pin) ADCx_IN6 . Channel different in bank A and bank B.

LL_ADC_CHANNEL_7	ADC external channel (channel connected to GPIO pin) ADCx_IN7 . Channel different in bank A and bank B.
LL_ADC_CHANNEL_8	ADC external channel (channel connected to GPIO pin) ADCx_IN8 . Channel different in bank A and bank B.
LL_ADC_CHANNEL_9	ADC external channel (channel connected to GPIO pin) ADCx_IN9 . Channel different in bank A and bank B.
LL_ADC_CHANNEL_10	ADC external channel (channel connected to GPIO pin) ADCx_IN10. Channel different in bank A and bank B.
LL_ADC_CHANNEL_11	ADC external channel (channel connected to GPIO pin) ADCx_IN11. Channel different in bank A and bank B.
LL_ADC_CHANNEL_12	ADC external channel (channel connected to GPIO pin) ADCx_IN12. Channel different in bank A and bank B.
LL_ADC_CHANNEL_13	ADC external channel (channel connected to GPIO pin) ADCx_IN13. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_14	ADC external channel (channel connected to GPIO pin) ADCx_IN14. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_15	ADC external channel (channel connected to GPIO pin) ADCx_IN15. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_16	ADC external channel (channel connected to GPIO pin) ADCx_IN16. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_17	ADC external channel (channel connected to GPIO pin) ADCx_IN17. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_18	ADC external channel (channel connected to GPIO pin) ADCx_IN18. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_19	ADC external channel (channel connected to GPIO pin) ADCx_IN19. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_20	ADC external channel (channel connected to GPIO pin) ADCx_IN20. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_21	ADC external channel (channel connected to GPIO pin) ADCx_IN21. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_22	ADC external channel (channel connected to GPIO pin) ADCx_IN22. Direct (fast) channel.

LL_ADC_CHANNEL_23	ADC external channel (channel connected to GPIO pin) ADCx_IN23. Direct (fast) channel.
LL_ADC_CHANNEL_24	ADC external channel (channel connected to GPIO pin) ADCx_IN24. Direct (fast) channel.
LL_ADC_CHANNEL_25	ADC external channel (channel connected to GPIO pin) ADCx_IN25. Direct (fast) channel.
LL_ADC_CHANNEL_26	ADC external channel (channel connected to GPIO pin) ADCx_IN26. Direct (fast) channel.
LL_ADC_CHANNEL_27	ADC external channel (channel connected to GPIO pin) ADCx_IN27. Channel common to both bank A and bank B. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_CHANNEL_28	ADC external channel (channel connected to GPIO pin) ADCx_IN28. Channel common to both bank A and bank B. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_CHANNEL_29	ADC external channel (channel connected to GPIO pin) ADCx_IN29. Channel common to both bank A and bank B. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_CHANNEL_30	ADC external channel (channel connected to GPIO pin) ADCx_IN30. Channel common to both bank A and bank B. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_CHANNEL_31	ADC external channel (channel connected to GPIO pin) ADCx_IN31. Channel common to both bank A and bank B. On STM32L1, parameter not available on all devices: only on STM32L1 Cat.4 and Cat.5.
LL_ADC_CHANNEL_VREFINT	ADC internal channel connected to VrefInt: Internal voltage reference. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_TEMPSENSOR	ADC internal channel connected to Temperature sensor. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_VCOMP	ADC internal channel connected to comparator COMP1 positive input via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_VOPAMP1	ADC internal channel connected to OPAMP1 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_VOPAMP2	ADC internal channel connected to OPAMP2 output via ADC switch matrix. Channel common to both bank A and bank B.
LL_ADC_CHANNEL_VOPAMP3	ADC internal channel connected to OPAMP3 output via ADC switch matrix. Channel common to both bank A and bank B.

ADC instance - Channels bank

<code>LL_ADC_CHANNELS_BANK_A</code>	ADC channels bank A
<code>LL_ADC_CHANNELS_BANK_B</code>	ADC channels bank B, available in devices categories 3, 4, 5.

Channel - Routing channels list

<code>LL_ADC_CHANNEL_3_ROUTING</code>	ADC channel 3 routing. Used as ADC direct channel (fast channel) if OPAMP1 is in power down mode.
<code>LL_ADC_CHANNEL_8_ROUTING</code>	ADC channel 8 routing. Used as ADC direct channel (fast channel) if OPAMP2 is in power down mode.
<code>LL_ADC_CHANNEL_13_ROUTING</code>	ADC channel 13 routing. Used as ADC re-routed channel if OPAMP3 is in power down mode. Otherwise, channel 13 is connected to OPAMP3 output and routed through switches COMP1_SW1 and VCOMP to ADC switch matrix. (Note: OPAMP3 is available on STM32L1 Cat.4 only).

Channel - Routing selection

<code>LL_ADC_CHANNEL_ROUTING_DEFAULT</code>	ADC channel routing default: slow channel
<code>LL_ADC_CHANNEL_ROUTING_DIRECT</code>	ADC channel routing direct: fast channel.

Channel - Sampling time

<code>LL_ADC_SAMPLINGTIME_4CYCLES</code>	Sampling time 4 ADC clock cycles
<code>LL_ADC_SAMPLINGTIME_9CYCLES</code>	Sampling time 9 ADC clock cycles
<code>LL_ADC_SAMPLINGTIME_16CYCLES</code>	Sampling time 16 ADC clock cycles
<code>LL_ADC_SAMPLINGTIME_24CYCLES</code>	Sampling time 24 ADC clock cycles
<code>LL_ADC_SAMPLINGTIME_48CYCLES</code>	Sampling time 48 ADC clock cycles
<code>LL_ADC_SAMPLINGTIME_96CYCLES</code>	Sampling time 96 ADC clock cycles
<code>LL_ADC_SAMPLINGTIME_192CYCLES</code>	Sampling time 192 ADC clock cycles
<code>LL_ADC_SAMPLINGTIME_384CYCLES</code>	Sampling time 384 ADC clock cycles

ADC common - Clock source

<code>LL_ADC_CLOCK_ASYNC_DIV1</code>	ADC asynchronous clock without prescaler
<code>LL_ADC_CLOCK_ASYNC_DIV2</code>	ADC asynchronous clock with prescaler division by 2
<code>LL_ADC_CLOCK_ASYNC_DIV4</code>	ADC asynchronous clock with prescaler division by 4

ADC common - Measurement path to internal channels

<code>LL_ADC_PATH_INTERNAL_NONE</code>	ADC measurement paths all disabled
<code>LL_ADC_PATH_INTERNAL_VREFINT</code>	ADC measurement path to internal channel VrefInt
<code>LL_ADC_PATH_INTERNAL_TEMPSENSOR</code>	ADC measurement path to internal channel temperature sensor

ADC instance - Data alignment

<code>LL_ADC_DATA_ALIGN_RIGHT</code>	ADC conversion data alignment: right aligned (alignment on data register LSB bit 0)
<code>LL_ADC_DATA_ALIGN_LEFT</code>	ADC conversion data alignment: left aligned (alignment

on data register MSB bit 15)

ADC flags

LL_ADC_FLAG_ADRDY	ADC flag ADC instance ready
LL_ADC_FLAG_STRT	ADC flag ADC group regular conversion start
LL_ADC_FLAG_EOCS	ADC flag ADC group regular end of unitary conversion or sequence conversions (to configure flag of end of conversion, use function)
LL_ADC_FLAG_OVR	ADC flag ADC group regular overrun
LL_ADC_FLAG_JSTRT	ADC flag ADC group injected conversion start
LL_ADC_FLAG_JEOS	ADC flag ADC group injected end of sequence conversions (Note: on this STM32 serie, there is no flag ADC group injected end of unitary conversion. Flag noted as "JEOS" is corresponding to flag "JEOS" in other STM32 families)
LL_ADC_FLAG_AWD1	ADC flag ADC analog watchdog 1

ADC instance - Groups

LL_ADC_GROUP_REGULAR	ADC group regular (available on all STM32 devices)
LL_ADC_GROUP_INJECTED	ADC group injected (not available on all STM32 devices)
LL_ADC_GROUP_REGULAR_INJECTED	ADC both groups regular and injected

Definitions of ADC hardware constraints delays

LL_ADC_DELAY_VREFINT_STAB_US	Delay for internal voltage reference stabilization time
LL_ADC_DELAY_TEMPSENSOR_STAB_US	Delay for internal voltage reference stabilization time

ADC group injected - Sequencer discontinuous mode

LL_ADC_INJ_SEQ_DISCONT_DISABLE	ADC group injected sequencer discontinuous mode disable
LL_ADC_INJ_SEQ_DISCONT_1RANK	ADC group injected sequencer discontinuous mode enable with sequence interruption every rank

ADC group injected - Sequencer ranks

LL_ADC_INJ_RANK_1	ADC group injected sequencer rank 1
LL_ADC_INJ_RANK_2	ADC group injected sequencer rank 2
LL_ADC_INJ_RANK_3	ADC group injected sequencer rank 3
LL_ADC_INJ_RANK_4	ADC group injected sequencer rank 4

ADC group injected - Sequencer scan length

LL_ADC_INJ_SEQ_SCAN_DISABLE	ADC group injected sequencer disable (equivalent to sequencer of 1 rank: ADC conversion on only 1 channel)
LL_ADC_INJ_SEQ_SCAN_ENABLE_2RANKS	ADC group injected sequencer enable

		with 2 ranks in the sequence
LL_ADC_INJ_SEQ_SCAN_ENABLE_3RANKS		ADC group injected sequencer enable with 3 ranks in the sequence
LL_ADC_INJ_SEQ_SCAN_ENABLE_4RANKS		ADC group injected sequencer enable with 4 ranks in the sequence
<i>ADC group injected - Trigger edge</i>		
LL_ADC_INJ_TRIG_EXT_RISING		ADC group injected conversion trigger polarity set to rising edge
LL_ADC_INJ_TRIG_EXT_FALLING		ADC group injected conversion trigger polarity set to falling edge
LL_ADC_INJ_TRIG_EXT_RISINGFALLING		ADC group injected conversion trigger polarity set to both rising and falling edges
<i>ADC group injected - Trigger source</i>		
LL_ADC_INJ_TRIG_SOFTWARE		ADC group injected conversion trigger internal: SW start.
LL_ADC_INJ_TRIG_EXT_TIM9_CH1		ADC group injected conversion trigger from external IP: TIM9 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM9_TRGO		ADC group injected conversion trigger from external IP: TIM9 TRGO. Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM2_TRGO		ADC group injected conversion trigger from external IP: TIM2 TRGO. Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM2_CH1		ADC group injected conversion trigger from external IP: TIM2 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM3_CH4		ADC group injected conversion trigger from external IP: TIM3 channel 4 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM4_TRGO		ADC group injected conversion trigger from external IP: TIM4 TRGO. Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM4_CH1		ADC group injected conversion trigger from external IP: TIM4 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM4_CH2		ADC group injected conversion trigger from external IP: TIM4 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM4_CH3		ADC group injected conversion trigger from external IP: TIM4 channel 3 event (capture compare: input capture or output capture).

	Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM10_CH1	ADC group injected conversion trigger from external IP: TIM10 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT_TIM7_TRGO	ADC group injected conversion trigger from external IP: TIM7 TRGO. Trigger edge set to rising edge (default setting).
LL_ADC_INJ_TRIG_EXT EXTI_LINE15	ADC group injected conversion trigger from external IP: external interrupt line 15. Trigger edge set to rising edge (default setting).

ADC group injected - Automatic trigger mode

LL_ADC_INJ_TRIG_INDEPENDENT	ADC group injected conversion trigger independent. Setting mandatory if ADC group injected injected trigger source is set to an external trigger.
LL_ADC_INJ_TRIG_FROM_GRP_REGULAR	ADC group injected conversion trigger from ADC group regular. Setting compliant only with group injected trigger source set to SW start, without any further action on ADC group injected conversion start or stop: in this case, ADC group injected is controlled only from ADC group regular.

ADC interruptions for configuration (interruption enable or disable)

LL_ADC_IT_EOCS	ADC interruption ADC group regular end of unitary conversion or sequence conversions (to configure flag of end of conversion, use function
LL_ADC_IT_OVR	ADC interruption ADC group regular overrun
LL_ADC_IT_JEOS	ADC interruption ADC group injected end of sequence conversions (Note: on this STM32 serie, there is no flag ADC group injected end of unitary conversion. Flag noted as "JEOC" is corresponding to flag "JEOS" in other STM32 families)
LL_ADC_IT_AWD1	ADC interruption ADC analog watchdog 1

ADC instance - Low power mode auto power-off

LL_ADC_LP_AUTOPOWEROFF_NONE	ADC low power mode auto power-off not activated
LL_ADC_LP_AUTOPOWEROFF_IDLE_PHASE	ADC low power mode auto power-off: ADC power off when ADC is not converting (idle phase)
LL_ADC_LP_AUTOPOWEROFF_AUTOWAIT_PHASE	ADC low power mode auto power-off: ADC power off when a delay is inserted between conversions (refer to

	function
LL_ADC_LP_AUTOPOWEROFF_IDLE_AUTOWAIT_PHASES	ADC low power mode auto power-off: ADC power off when ADC is not converting (idle phase) and when a delay is inserted between conversions (refer to function)
<i>ADC instance - Low power mode auto wait (auto delay)</i>	
LL_ADC_LP_AUTOWAIT_NONE	ADC low power mode auto wait not activated
LL_ADC_LP_AUTOWAIT	ADC low power mode auto wait: Dynamic low power mode, ADC conversions are performed only when necessary (when previous ADC conversion data is read). See description with function
LL_ADC_LP_AUTOWAIT_7_APBCLOCKCYCLES	ADC low power mode auto wait: Insert a delay between ADC conversions: 7 APB clock cycles
LL_ADC_LP_AUTOWAIT_15_APBCLOCKCYCLES	ADC low power mode auto wait: Insert a delay between ADC conversions: 15 APB clock cycles
LL_ADC_LP_AUTOWAIT_31_APBCLOCKCYCLES	ADC low power mode auto wait: Insert a delay between ADC conversions: 31 APB clock cycles
LL_ADC_LP_AUTOWAIT_63_APBCLOCKCYCLES	ADC low power mode auto wait: Insert a delay between ADC conversions: 63 APB clock cycles
LL_ADC_LP_AUTOWAIT_127_APBCLOCKCYCLES	ADC low power mode auto wait: Insert a delay between ADC conversions: 127 APB clock cycles
LL_ADC_LP_AUTOWAIT_255_APBCLOCKCYCLES	ADC low power mode auto wait: Insert a delay between ADC conversions: 255 APB clock cycles

ADC registers compliant with specific purpose

LL_ADC_DMA_REG_REGULAR_DATA

ADC group regular - Continuous mode

LL_ADC_REG_CONV_SINGLE	ADC conversions are performed in single mode: one conversion per trigger
LL_ADC_REG_CONV_CONTINUOUS	ADC conversions are performed in continuous mode: after the first trigger, following conversions launched successively automatically

ADC group regular - DMA transfer of ADC conversion data

LL_ADC_REG_DMA_TRANSFER_NONE ADC conversions are not transferred by

	DMA
LL_ADC_REG_DMA_TRANSFER_LIMITED	ADC conversion data are transferred by DMA, in limited mode (one shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular.
LL_ADC_REG_DMA_TRANSFER_UNLIMITED	ADC conversion data are transferred by DMA, in unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transferred (number of ADC conversions). This ADC mode is intended to be used with DMA mode circular.
ADC group regular - Flag EOC selection (unitary or sequence conversions)	
LL_ADC_REG_FLAG_EOC_SEQUENCE_CONV	ADC flag EOC (end of unitary conversion) selected
LL_ADC_REG_FLAG_EOC_UNITARY_CONV	ADC flag EOS (end of sequence conversions) selected
ADC group regular - Sequencer discontinuous mode	
LL_ADC_REG_SEQ_DISCONT_DISABLE	ADC group regular sequencer discontinuous mode disable
LL_ADC_REG_SEQ_DISCONT_1RANK	ADC group regular sequencer discontinuous mode enable with sequence interruption every rank
LL_ADC_REG_SEQ_DISCONT_2RANKS	ADC group regular sequencer discontinuous mode enabled with sequence interruption every 2 ranks
LL_ADC_REG_SEQ_DISCONT_3RANKS	ADC group regular sequencer discontinuous mode enable with sequence interruption every 3 ranks
LL_ADC_REG_SEQ_DISCONT_4RANKS	ADC group regular sequencer discontinuous mode enable with sequence interruption every 4 ranks
LL_ADC_REG_SEQ_DISCONT_5RANKS	ADC group regular sequencer discontinuous mode enable with sequence interruption every 5 ranks
LL_ADC_REG_SEQ_DISCONT_6RANKS	ADC group regular sequencer discontinuous mode enable with sequence interruption every 6 ranks
LL_ADC_REG_SEQ_DISCONT_7RANKS	ADC group regular sequencer discontinuous mode enable with sequence interruption every 7 ranks
LL_ADC_REG_SEQ_DISCONT_8RANKS	ADC group regular sequencer discontinuous mode enable with sequence interruption every 8 ranks

ADC group regular - Sequencer ranks

LL_ADC_REG_RANK_1	ADC group regular sequencer rank 1
LL_ADC_REG_RANK_2	ADC group regular sequencer rank 2
LL_ADC_REG_RANK_3	ADC group regular sequencer rank 3
LL_ADC_REG_RANK_4	ADC group regular sequencer rank 4
LL_ADC_REG_RANK_5	ADC group regular sequencer rank 5
LL_ADC_REG_RANK_6	ADC group regular sequencer rank 6
LL_ADC_REG_RANK_7	ADC group regular sequencer rank 7
LL_ADC_REG_RANK_8	ADC group regular sequencer rank 8
LL_ADC_REG_RANK_9	ADC group regular sequencer rank 9
LL_ADC_REG_RANK_10	ADC group regular sequencer rank 10
LL_ADC_REG_RANK_11	ADC group regular sequencer rank 11
LL_ADC_REG_RANK_12	ADC group regular sequencer rank 12
LL_ADC_REG_RANK_13	ADC group regular sequencer rank 13
LL_ADC_REG_RANK_14	ADC group regular sequencer rank 14
LL_ADC_REG_RANK_15	ADC group regular sequencer rank 15
LL_ADC_REG_RANK_16	ADC group regular sequencer rank 16
LL_ADC_REG_RANK_17	ADC group regular sequencer rank 17
LL_ADC_REG_RANK_18	ADC group regular sequencer rank 18
LL_ADC_REG_RANK_19	ADC group regular sequencer rank 19
LL_ADC_REG_RANK_20	ADC group regular sequencer rank 20
LL_ADC_REG_RANK_21	ADC group regular sequencer rank 21
LL_ADC_REG_RANK_22	ADC group regular sequencer rank 22
LL_ADC_REG_RANK_23	ADC group regular sequencer rank 23
LL_ADC_REG_RANK_24	ADC group regular sequencer rank 24
LL_ADC_REG_RANK_25	ADC group regular sequencer rank 25
LL_ADC_REG_RANK_26	ADC group regular sequencer rank 26
LL_ADC_REG_RANK_27	ADC group regular sequencer rank 27
LL_ADC_REG_RANK_28	ADC group regular sequencer rank 28

ADC group regular - Sequencer scan length

LL_ADC_REG_SEQ_SCAN_DISABLE	ADC group regular sequencer disable (equivalent to sequencer of 1 rank: ADC conversion on only 1 channel)
LL_ADC_REG_SEQ_SCAN_ENABLE_2RANKS	ADC group regular sequencer enable with 2 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_3RANKS	ADC group regular sequencer enable with 3 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_4RANKS	ADC group regular sequencer enable

LL_ADC_REG_SEQ_SCAN_ENABLE_5RANKS	with 4 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_6RANKS	ADC group regular sequencer enable with 5 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_7RANKS	ADC group regular sequencer enable with 6 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_8RANKS	ADC group regular sequencer enable with 7 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_9RANKS	ADC group regular sequencer enable with 8 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_10RANKS	ADC group regular sequencer enable with 9 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_11RANKS	ADC group regular sequencer enable with 10 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_12RANKS	ADC group regular sequencer enable with 11 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_13RANKS	ADC group regular sequencer enable with 12 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_14RANKS	ADC group regular sequencer enable with 13 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_15RANKS	ADC group regular sequencer enable with 14 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_16RANKS	ADC group regular sequencer enable with 15 ranks in the sequence
LL_ADC_REG_SEQ_SCAN_ENABLE_17RANKS	ADC group regular sequencer enable with 16 ranks in the sequence

ADC group regular - Trigger edge

LL_ADC_REG_TRIG_EXT_RISING	ADC group regular conversion trigger polarity set to rising edge
LL_ADC_REG_TRIG_EXT_FALLING	ADC group regular conversion trigger polarity set to falling edge
LL_ADC_REG_TRIG_EXT_RISINGFALLING	ADC group regular conversion trigger polarity set to both rising and falling edges

ADC group regular - Trigger source

LL_ADC_REG_TRIG_SOFTWARE	ADC group regular conversion trigger internal: SW start.
LL_ADC_REG_TRIG_EXT_TIM2_TRGO	ADC group regular conversion trigger from external IP: TIM2 TRGO. Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM2_CH3	ADC group regular conversion trigger from external IP: TIM2 channel 3 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM3_TRGO	ADC group regular conversion trigger from external IP: TIM3 TRGO. Trigger edge set to

	rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM2_CH2	ADC group regular conversion trigger from external IP: TIM2 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM3_CH1	ADC group regular conversion trigger from external IP: TIM3 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM3_CH3	ADC group regular conversion trigger from external IP: TIM3 channel 3 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM4_TRGO	ADC group regular conversion trigger from external IP: TIM4 TRGO. Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM4_CH4	ADC group regular conversion trigger from external IP: TIM4 channel 4 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM6_TRGO	ADC group regular conversion trigger from external IP: TIM6 TRGO. Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM9_CH2	ADC group regular conversion trigger from external IP: TIM9 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT_TIM9_TRGO	ADC group regular conversion trigger from external IP: TIM9 TRGO. Trigger edge set to rising edge (default setting).
LL_ADC_REG_TRIG_EXT EXTI_LINE11	ADC group regular conversion trigger from external IP: external interrupt line 11. Trigger edge set to rising edge (default setting).

ADC instance - Resolution

LL_ADC_RESOLUTION_12B	ADC resolution 12 bits
LL_ADC_RESOLUTION_10B	ADC resolution 10 bits
LL_ADC_RESOLUTION_8B	ADC resolution 8 bits
LL_ADC_RESOLUTION_6B	ADC resolution 6 bits

ADC instance - Scan selection

LL_ADC_SEQ_SCAN_DISABLE	ADC conversion is performed in unitary conversion mode (one channel converted, that defined in rank 1). Configuration of both groups regular and injected
-------------------------	---

sequencers (sequence length, ...) is discarded:
equivalent to length of 1 rank.

LL_ADC_SEQ_SCAN_ENABLE

ADC conversions are performed in sequence conversions mode, according to configuration of both groups regular and injected sequencers (sequence length, ...).

ADC helper macro

_LL_ADC_CHANNEL_TO_DECI MAL_NB

Description:

- Helper macro to get ADC channel number in decimal format from literals
LL_ADC_CHANNEL_x.

Parameters:

- _CHANNEL_: This parameter can be one of the following values:
 - LL_ADC_CHANNEL_0 (2)
 - LL_ADC_CHANNEL_1 (2)
 - LL_ADC_CHANNEL_2 (2)
 - LL_ADC_CHANNEL_3 (2)
 - LL_ADC_CHANNEL_4 (1)
 - LL_ADC_CHANNEL_5 (1)
 - LL_ADC_CHANNEL_6 (2)
 - LL_ADC_CHANNEL_7 (2)
 - LL_ADC_CHANNEL_8 (2)
 - LL_ADC_CHANNEL_9 (2)
 - LL_ADC_CHANNEL_10 (2)
 - LL_ADC_CHANNEL_11 (2)
 - LL_ADC_CHANNEL_12 (2)
 - LL_ADC_CHANNEL_13 (3)
 - LL_ADC_CHANNEL_14 (3)
 - LL_ADC_CHANNEL_15 (3)
 - LL_ADC_CHANNEL_16 (3)
 - LL_ADC_CHANNEL_17 (3)
 - LL_ADC_CHANNEL_18 (3)
 - LL_ADC_CHANNEL_19 (3)
 - LL_ADC_CHANNEL_20 (3)
 - LL_ADC_CHANNEL_21 (3)
 - LL_ADC_CHANNEL_22 (1)
 - LL_ADC_CHANNEL_23 (1)
 - LL_ADC_CHANNEL_24 (1)
 - LL_ADC_CHANNEL_25 (1)
 - LL_ADC_CHANNEL_26 (3)
 - LL_ADC_CHANNEL_27 (3)(4)
 - LL_ADC_CHANNEL_28 (3)(4)
 - LL_ADC_CHANNEL_29 (3)(4)
 - LL_ADC_CHANNEL_30 (3)(4)
 - LL_ADC_CHANNEL_31 (3)(4)
 - LL_ADC_CHANNEL_VREFINT (3)
 - LL_ADC_CHANNEL_TEMPSENSOR (3)
 - LL_ADC_CHANNEL_VCOMP (3)
 - LL_ADC_CHANNEL_VOPAMP1 (3)(5)

- LL_ADC_CHANNEL_VOPAMP2 (3)(5)
- LL_ADC_CHANNEL_VOPAMP3 (3)(5)

Return value:

- Value: between Min_Data=0 and Max_Data=18

Notes:

- Example:
`_LL_ADC_CHANNEL_TO_DECIMAL_NB(LL_ADC_CHANNEL_4)` will return decimal number "4". The input can be a value from functions where a channel number is returned, either defined with number or with bitfield (only one bit must be set).

<code>_LL_ADC_DECIMAL_NB_TO_CH ANNEL</code>	Description:
---	---------------------

- Helper macro to get ADC channel in literal format `LL_ADC_CHANNEL_x` from number in decimal format.

Parameters:

- `_DECIMAL_NB_`: Value between Min_Data=0 and Max_Data=18

Return value:

- Returned: value can be one of the following values:
 - `LL_ADC_CHANNEL_0` (2)
 - `LL_ADC_CHANNEL_1` (2)
 - `LL_ADC_CHANNEL_2` (2)
 - `LL_ADC_CHANNEL_3` (2)
 - `LL_ADC_CHANNEL_4` (1)
 - `LL_ADC_CHANNEL_5` (1)
 - `LL_ADC_CHANNEL_6` (2)
 - `LL_ADC_CHANNEL_7` (2)
 - `LL_ADC_CHANNEL_8` (2)
 - `LL_ADC_CHANNEL_9` (2)
 - `LL_ADC_CHANNEL_10` (2)
 - `LL_ADC_CHANNEL_11` (2)
 - `LL_ADC_CHANNEL_12` (2)
 - `LL_ADC_CHANNEL_13` (3)
 - `LL_ADC_CHANNEL_14` (3)
 - `LL_ADC_CHANNEL_15` (3)
 - `LL_ADC_CHANNEL_16` (3)
 - `LL_ADC_CHANNEL_17` (3)
 - `LL_ADC_CHANNEL_18` (3)
 - `LL_ADC_CHANNEL_19` (3)
 - `LL_ADC_CHANNEL_20` (3)
 - `LL_ADC_CHANNEL_21` (3)
 - `LL_ADC_CHANNEL_22` (1)
 - `LL_ADC_CHANNEL_23` (1)
 - `LL_ADC_CHANNEL_24` (1)
 - `LL_ADC_CHANNEL_25` (1)

- LL_ADC_CHANNEL_26 (3)
- LL_ADC_CHANNEL_27 (3)(4)
- LL_ADC_CHANNEL_28 (3)(4)
- LL_ADC_CHANNEL_29 (3)(4)
- LL_ADC_CHANNEL_30 (3)(4)
- LL_ADC_CHANNEL_31 (3)(4)
- LL_ADC_CHANNEL_VREFINT (3)(6)
- LL_ADC_CHANNEL_TEMPSENSOR (3)(6)
- LL_ADC_CHANNEL_VCOMP (3)(6)
- LL_ADC_CHANNEL_VOPAMP1 (3)(5)
- LL_ADC_CHANNEL_VOPAMP2 (3)(5)
- LL_ADC_CHANNEL_VOPAMP3 (3)(5)

Notes:

- Example:
`_LL_ADC_DECIMAL_NB_TO_CHANNEL(4)`
 will return a data equivalent to
`"LL_ADC_CHANNEL_4"`.

[LL_ADC_IS_CHANNEL_INTERNAL](#)**Description:**

- Helper macro to determine whether the selected channel corresponds to literal definitions of driver.

Parameters:

- `_CHANNEL_`: This parameter can be one of the following values:
 - LL_ADC_CHANNEL_0 (2)
 - LL_ADC_CHANNEL_1 (2)
 - LL_ADC_CHANNEL_2 (2)
 - LL_ADC_CHANNEL_3 (2)
 - LL_ADC_CHANNEL_4 (1)
 - LL_ADC_CHANNEL_5 (1)
 - LL_ADC_CHANNEL_6 (2)
 - LL_ADC_CHANNEL_7 (2)
 - LL_ADC_CHANNEL_8 (2)
 - LL_ADC_CHANNEL_9 (2)
 - LL_ADC_CHANNEL_10 (2)
 - LL_ADC_CHANNEL_11 (2)
 - LL_ADC_CHANNEL_12 (2)
 - LL_ADC_CHANNEL_13 (3)
 - LL_ADC_CHANNEL_14 (3)
 - LL_ADC_CHANNEL_15 (3)
 - LL_ADC_CHANNEL_16 (3)
 - LL_ADC_CHANNEL_17 (3)
 - LL_ADC_CHANNEL_18 (3)
 - LL_ADC_CHANNEL_19 (3)
 - LL_ADC_CHANNEL_20 (3)
 - LL_ADC_CHANNEL_21 (3)
 - LL_ADC_CHANNEL_22 (1)
 - LL_ADC_CHANNEL_23 (1)
 - LL_ADC_CHANNEL_24 (1)

- LL_ADC_CHANNEL_25 (1)
- LL_ADC_CHANNEL_26 (3)
- LL_ADC_CHANNEL_27 (3)(4)
- LL_ADC_CHANNEL_28 (3)(4)
- LL_ADC_CHANNEL_29 (3)(4)
- LL_ADC_CHANNEL_30 (3)(4)
- LL_ADC_CHANNEL_31 (3)(4)
- LL_ADC_CHANNEL_VREFINT (3)
- LL_ADC_CHANNEL_TEMPSENSOR (3)
- LL_ADC_CHANNEL_VCOMP (3)
- LL_ADC_CHANNEL_VOPAMP1 (3)(5)
- LL_ADC_CHANNEL_VOPAMP2 (3)(5)
- LL_ADC_CHANNEL_VOPAMP3 (3)(5)

Return value:

- Value: "0" if the channel corresponds to a parameter definition of a ADC external channel (channel connected to a GPIO pin). Value "1" if the channel corresponds to a parameter definition of a ADC internal channel.

Notes:

- The different literal definitions of ADC channels are: ADC internal channel:
LL_ADC_CHANNEL_VREFINT,
LL_ADC_CHANNEL_TEMPSENSOR, ...ADC external channel (channel connected to a GPIO pin): LL_ADC_CHANNEL_1,
LL_ADC_CHANNEL_2, ... The channel parameter must be a value defined from literal definition of a ADC internal channel (LL_ADC_CHANNEL_VREFINT,
LL_ADC_CHANNEL_TEMPSENSOR, ...), ADC external channel (LL_ADC_CHANNEL_1,
LL_ADC_CHANNEL_2, ...), must not be a value from functions where a channel number is returned from ADC registers, because internal and external channels share the same channel number in ADC registers. The differentiation is made only with parameters definitions of driver.

`_LL_ADC_CHANNEL_INTERNAL
_TO_EXTERNAL`

Description:

- Helper macro to convert a channel defined from parameter definition of a ADC internal channel (LL_ADC_CHANNEL_VREFINT,
LL_ADC_CHANNEL_TEMPSENSOR, ...), to its equivalent parameter definition of a ADC external channel (LL_ADC_CHANNEL_1,
LL_ADC_CHANNEL_2, ...).

Parameters:

- `_CHANNEL_`: This parameter can be one of the following values:
 - LL_ADC_CHANNEL_0 (2)

- LL_ADC_CHANNEL_1 (2)
- LL_ADC_CHANNEL_2 (2)
- LL_ADC_CHANNEL_3 (2)
- LL_ADC_CHANNEL_4 (1)
- LL_ADC_CHANNEL_5 (1)
- LL_ADC_CHANNEL_6 (2)
- LL_ADC_CHANNEL_7 (2)
- LL_ADC_CHANNEL_8 (2)
- LL_ADC_CHANNEL_9 (2)
- LL_ADC_CHANNEL_10 (2)
- LL_ADC_CHANNEL_11 (2)
- LL_ADC_CHANNEL_12 (2)
- LL_ADC_CHANNEL_13 (3)
- LL_ADC_CHANNEL_14 (3)
- LL_ADC_CHANNEL_15 (3)
- LL_ADC_CHANNEL_16 (3)
- LL_ADC_CHANNEL_17 (3)
- LL_ADC_CHANNEL_18 (3)
- LL_ADC_CHANNEL_19 (3)
- LL_ADC_CHANNEL_20 (3)
- LL_ADC_CHANNEL_21 (3)
- LL_ADC_CHANNEL_22 (1)
- LL_ADC_CHANNEL_23 (1)
- LL_ADC_CHANNEL_24 (1)
- LL_ADC_CHANNEL_25 (1)
- LL_ADC_CHANNEL_26 (3)
- LL_ADC_CHANNEL_27 (3)(4)
- LL_ADC_CHANNEL_28 (3)(4)
- LL_ADC_CHANNEL_29 (3)(4)
- LL_ADC_CHANNEL_30 (3)(4)
- LL_ADC_CHANNEL_31 (3)(4)
- LL_ADC_CHANNEL_VREFINT (3)
- LL_ADC_CHANNEL_TEMPSENSOR (3)
- LL_ADC_CHANNEL_VCOMP (3)
- LL_ADC_CHANNEL_VOPAMP1 (3)(5)
- LL_ADC_CHANNEL_VOPAMP2 (3)(5)
- LL_ADC_CHANNEL_VOPAMP3 (3)(5)

Return value:

- Returned: value can be one of the following values:
 - LL_ADC_CHANNEL_0
 - LL_ADC_CHANNEL_1
 - LL_ADC_CHANNEL_2
 - LL_ADC_CHANNEL_3
 - LL_ADC_CHANNEL_4
 - LL_ADC_CHANNEL_5
 - LL_ADC_CHANNEL_6
 - LL_ADC_CHANNEL_7
 - LL_ADC_CHANNEL_8
 - LL_ADC_CHANNEL_9
 - LL_ADC_CHANNEL_10
 - LL_ADC_CHANNEL_11

- LL_ADC_CHANNEL_12
- LL_ADC_CHANNEL_13
- LL_ADC_CHANNEL_14
- LL_ADC_CHANNEL_15
- LL_ADC_CHANNEL_16
- LL_ADC_CHANNEL_17
- LL_ADC_CHANNEL_18

Notes:

- The channel parameter can be, additionally to a value defined from parameter definition of a ADC internal channel (LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, ...), a value defined from parameter definition of ADC external channel (LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, ...) or a value from functions where a channel number is returned from ADC registers.

_LL_ADC_IS_CHANNEL_INTERNAL_AVAILABLE**Description:**

- Helper macro to determine whether the internal channel selected is available on the ADC instance selected.

Parameters:

- _ADC_INSTANCE_: ADC instance
- _CHANNEL_: This parameter can be one of the following values:
 - LL_ADC_CHANNEL_VREFINT (3)
 - LL_ADC_CHANNEL_TEMPSENSOR (3)
 - LL_ADC_CHANNEL_VCOMP (3)
 - LL_ADC_CHANNEL_VOPAMP1 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP2 (3)(5)
 - LL_ADC_CHANNEL_VOPAMP3 (3)(5)

Return value:

- Value: "0" if the internal channel selected is not available on the ADC instance selected. Value "1" if the internal channel selected is available on the ADC instance selected.

Notes:

- The channel parameter must be a value defined from parameter definition of a ADC internal channel (LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, ...), must not be a value defined from parameter definition of ADC external channel (LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, ...) or a value from functions where a channel number is returned from ADC registers, because internal and external channels share the same channel

number in ADC registers. The differentiation is made only with parameters definitions of driver.

`_LL_ADC_ANALOGWD_CHANN
EL_GROUP`

Description:

- Helper macro to define ADC analog watchdog parameter: define a single channel to monitor with analog watchdog from sequencer channel and groups definition.

Parameters:

- `_CHANNEL_`: This parameter can be one of the following values:
 - `LL_ADC_CHANNEL_0` (2)
 - `LL_ADC_CHANNEL_1` (2)
 - `LL_ADC_CHANNEL_2` (2)
 - `LL_ADC_CHANNEL_3` (2)
 - `LL_ADC_CHANNEL_4` (1)
 - `LL_ADC_CHANNEL_5` (1)
 - `LL_ADC_CHANNEL_6` (2)
 - `LL_ADC_CHANNEL_7` (2)
 - `LL_ADC_CHANNEL_8` (2)
 - `LL_ADC_CHANNEL_9` (2)
 - `LL_ADC_CHANNEL_10` (2)
 - `LL_ADC_CHANNEL_11` (2)
 - `LL_ADC_CHANNEL_12` (2)
 - `LL_ADC_CHANNEL_13` (3)
 - `LL_ADC_CHANNEL_14` (3)
 - `LL_ADC_CHANNEL_15` (3)
 - `LL_ADC_CHANNEL_16` (3)
 - `LL_ADC_CHANNEL_17` (3)
 - `LL_ADC_CHANNEL_18` (3)
 - `LL_ADC_CHANNEL_19` (3)
 - `LL_ADC_CHANNEL_20` (3)
 - `LL_ADC_CHANNEL_21` (3)
 - `LL_ADC_CHANNEL_22` (1)
 - `LL_ADC_CHANNEL_23` (1)
 - `LL_ADC_CHANNEL_24` (1)
 - `LL_ADC_CHANNEL_25` (1)
 - `LL_ADC_CHANNEL_26` (3)
 - `LL_ADC_CHANNEL_27` (3)(4)
 - `LL_ADC_CHANNEL_28` (3)(4)
 - `LL_ADC_CHANNEL_29` (3)(4)
 - `LL_ADC_CHANNEL_30` (3)(4)
 - `LL_ADC_CHANNEL_31` (3)(4)
 - `LL_ADC_CHANNEL_VREFINT` (3)(6)
 - `LL_ADC_CHANNEL_TEMPSENSOR` (3)(6)
 - `LL_ADC_CHANNEL_VCOMP` (3)(6)
 - `LL_ADC_CHANNEL_VOPAMP1` (3)(5)
 - `LL_ADC_CHANNEL_VOPAMP2` (3)(5)
 - `LL_ADC_CHANNEL_VOPAMP3` (3)(5)
- `_GROUP_`: This parameter can be one of the following values:

- LL_ADC_GROUP_REGULAR
- LL_ADC_GROUP_INJECTED
- LL_ADC_GROUP_REGULAR_INJECTED

Return value:

- Returned: value can be one of the following values:
 - LL_ADC_AWD_DISABLE
 - LL_ADC_AWD_ALL_CHANNELS_REG
 - LL_ADC_AWD_ALL_CHANNELS_INJ
 - LL_ADC_AWD_ALL_CHANNELS_REG_INJ
 - LL_ADC_AWD_CHANNEL_0_REG (2)
 - LL_ADC_AWD_CHANNEL_0_INJ (2)
 - LL_ADC_AWD_CHANNEL_0_REG_INJ (2)
 - LL_ADC_AWD_CHANNEL_1_REG (2)
 - LL_ADC_AWD_CHANNEL_1_INJ (2)
 - LL_ADC_AWD_CHANNEL_1_REG_INJ (2)
 - LL_ADC_AWD_CHANNEL_2_REG (2)
 - LL_ADC_AWD_CHANNEL_2_INJ (2)
 - LL_ADC_AWD_CHANNEL_2_REG_INJ (2)
 - LL_ADC_AWD_CHANNEL_3_REG (2)
 - LL_ADC_AWD_CHANNEL_3_INJ (2)
 - LL_ADC_AWD_CHANNEL_3_REG_INJ (2)
 - LL_ADC_AWD_CHANNEL_4_REG (1)
 - LL_ADC_AWD_CHANNEL_4_INJ (1)
 - LL_ADC_AWD_CHANNEL_4_REG_INJ (1)
 - LL_ADC_AWD_CHANNEL_5_REG (1)
 - LL_ADC_AWD_CHANNEL_5_INJ (1)
 - LL_ADC_AWD_CHANNEL_5_REG_INJ (1)
 - LL_ADC_AWD_CHANNEL_6_REG (2)
 - LL_ADC_AWD_CHANNEL_6_INJ (2)
 - LL_ADC_AWD_CHANNEL_6_REG_INJ (2)
 - LL_ADC_AWD_CHANNEL_7_REG (2)
 - LL_ADC_AWD_CHANNEL_7_INJ (2)
 - LL_ADC_AWD_CHANNEL_7_REG_INJ (2)
 - LL_ADC_AWD_CHANNEL_8_REG (2)
 - LL_ADC_AWD_CHANNEL_8_INJ (2)
 - LL_ADC_AWD_CHANNEL_8_REG_INJ (2)
 - LL_ADC_AWD_CHANNEL_9_REG (2)
 - LL_ADC_AWD_CHANNEL_9_INJ (2)
 - LL_ADC_AWD_CHANNEL_9_REG_INJ (2)
 - LL_ADC_AWD_CHANNEL_10_REG (2)

- LL_ADC_AWD_CHANNEL_10_INJ (2)
- LL_ADC_AWD_CHANNEL_10_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_11_REG (2)
- LL_ADC_AWD_CHANNEL_11_INJ (2)
- LL_ADC_AWD_CHANNEL_11_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_12_REG (2)
- LL_ADC_AWD_CHANNEL_12_INJ (2)
- LL_ADC_AWD_CHANNEL_12_REG_INJ (2)
- LL_ADC_AWD_CHANNEL_13_REG (3)
- LL_ADC_AWD_CHANNEL_13_INJ (3)
- LL_ADC_AWD_CHANNEL_13_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_14_REG (3)
- LL_ADC_AWD_CHANNEL_14_INJ (3)
- LL_ADC_AWD_CHANNEL_14_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_15_REG (3)
- LL_ADC_AWD_CHANNEL_15_INJ (3)
- LL_ADC_AWD_CHANNEL_15_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_16_REG (3)
- LL_ADC_AWD_CHANNEL_16_INJ (3)
- LL_ADC_AWD_CHANNEL_16_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_17_REG (3)
- LL_ADC_AWD_CHANNEL_17_INJ (3)
- LL_ADC_AWD_CHANNEL_17_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_18_REG (3)
- LL_ADC_AWD_CHANNEL_18_INJ (3)
- LL_ADC_AWD_CHANNEL_18_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_19_REG (3)
- LL_ADC_AWD_CHANNEL_19_INJ (3)
- LL_ADC_AWD_CHANNEL_19_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_20_REG (3)
- LL_ADC_AWD_CHANNEL_20_INJ (3)
- LL_ADC_AWD_CHANNEL_20_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_21_REG (3)
- LL_ADC_AWD_CHANNEL_21_INJ (3)
- LL_ADC_AWD_CHANNEL_21_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_22_REG (1)
- LL_ADC_AWD_CHANNEL_22_INJ (1)
- LL_ADC_AWD_CHANNEL_22_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_23_REG (1)
- LL_ADC_AWD_CHANNEL_23_INJ (1)

- LL_ADC_AWD_CHANNEL_23_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_24_REG (1)
- LL_ADC_AWD_CHANNEL_24_INJ (1)
- LL_ADC_AWD_CHANNEL_24_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_25_REG (1)
- LL_ADC_AWD_CHANNEL_25_INJ (1)
- LL_ADC_AWD_CHANNEL_25_REG_INJ (1)
- LL_ADC_AWD_CHANNEL_26_REG (3)
- LL_ADC_AWD_CHANNEL_26_INJ (3)
- LL_ADC_AWD_CHANNEL_26_REG_INJ (3)
- LL_ADC_AWD_CHANNEL_27_REG (3)(4)
- LL_ADC_AWD_CHANNEL_27_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_27_REG_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_28_REG (3)(4)
- LL_ADC_AWD_CHANNEL_28_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_28_REG_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_29_REG (3)(4)
- LL_ADC_AWD_CHANNEL_29_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_29_REG_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_30_REG (3)(4)
- LL_ADC_AWD_CHANNEL_30_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_30_REG_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_31_REG (3)(4)
- LL_ADC_AWD_CHANNEL_31_INJ (3)(4)
- LL_ADC_AWD_CHANNEL_31_REG_INJ (3)(4)
- LL_ADC_AWD_CH_VREFINT_REG (3)
- LL_ADC_AWD_CH_VREFINT_INJ (3)
- LL_ADC_AWD_CH_VREFINT_REG_INJ (3)
- LL_ADC_AWD_CH_TEMPSENSOR_REG (3)
- LL_ADC_AWD_CH_TEMPSENSOR_INJ (3)
- LL_ADC_AWD_CH_TEMPSENSOR_REG_INJ (3)
- LL_ADC_AWD_CH_VCOMP_REG (3)
- LL_ADC_AWD_CH_VCOMP_INJ (3)
- LL_ADC_AWD_CH_VCOMP_REG_INJ (3)
- LL_ADC_AWD_CH_VOPAMP1_REG (3)(5)
- LL_ADC_AWD_CH_VOPAMP1_INJ (3)(5)
- LL_ADC_AWD_CH_VOPAMP1_REG_INJ (3)(5)

- LL_ADC_AWD_CH_VOPAMP2_REG (3)(5)
- LL_ADC_AWD_CH_VOPAMP2_INJ (3)(5)
- LL_ADC_AWD_CH_VOPAMP2_REG_INJ (3)(5)
- LL_ADC_AWD_CH_VOPAMP3_REG (3)(5)
- LL_ADC_AWD_CH_VOPAMP3_INJ (3)(5)
- LL_ADC_AWD_CH_VOPAMP3_REG_INJ (3)(5)

Notes:

- To be used with function LL_ADC_SetAnalogWDMonitChannels(). Example:
LL_ADC_SetAnalogWDMonitChannels(ADC1,
LL_ADC_AWD1,
__LL_ADC_ANALOGWD_CHANNEL_GROUP(
LL_ADC_CHANNEL4,
LL_ADC_GROUP_REGULAR))

__LL_ADC_ANALOGWD_SET_TH
RESHOLD_RESOLUTION

Description:

- Helper macro to set the value of ADC analog watchdog threshold high or low in function of ADC resolution, when ADC resolution is different of 12 bits.

Parameters:

- __ADC_RESOLUTION__: This parameter can be one of the following values:
 - LL_ADC_RESOLUTION_12B
 - LL_ADC_RESOLUTION_10B
 - LL_ADC_RESOLUTION_8B
 - LL_ADC_RESOLUTION_6B
- __AWD_THRESHOLD__: Value between Min_Data=0x000 and Max_Data=0xFFFF

Return value:

- Value: between Min_Data=0x000 and Max_Data=0xFFFF

Notes:

- To be used with function LL_ADC_SetAnalogWDThresholds(). Example, with a ADC resolution of 8 bits, to set the value of analog watchdog threshold high (on 8 bits): LL_ADC_SetAnalogWDThresholds (<ADCx param>, __LL_ADC_ANALOGWD_SET_THRESHOLD_RESOLUTION(LL_ADC_RESOLUTION_8B, <threshold_value_8_bits>));

__LL_ADC_ANALOGWD_GET_TH
RESHOLD_RESOLUTION

Description:

- Helper macro to get the value of ADC analog

watchdog threshold high or low in function of ADC resolution, when ADC resolution is different of 12 bits.

Parameters:

- `__ADC_RESOLUTION__`: This parameter can be one of the following values:
 - `LL_ADC_RESOLUTION_12B`
 - `LL_ADC_RESOLUTION_10B`
 - `LL_ADC_RESOLUTION_8B`
 - `LL_ADC_RESOLUTION_6B`
- `__AWD_THRESHOLD_12_BITS__`: Value between `Min_Data=0x000` and `Max_Data=0xFFFF`

Return value:

- Value: between `Min_Data=0x000` and `Max_Data=0xFFFF`

Notes:

- To be used with function `LL_ADC_GetAnalogWDThresholds()`. Example, with a ADC resolution of 8 bits, to get the value of analog watchdog threshold high (on 8 bits): <code><threshold_value_6_bits> = LL_ADC_ANALOGWD_GET_THRESHOLD_RESOLUTION(LL_ADC_RESOLUTION_8B, LL_ADC_GetAnalogWDThresholds(<ADCx param>, LL_ADC_AWD_THRESHOLD_HIGH));</code>

`__LL_ADC_COMMON_INSTANCE`

Description:

- Helper macro to select the ADC common instance to which is belonging the selected ADC instance.

Parameters:

- `__ADCx__`: ADC instance

Return value:

- ADC: common register instance

Notes:

- ADC common register instance can be used for: Set parameters common to several ADC instancesMultimode (for devices with several ADC instances) Refer to functions having argument "ADCxy_COMMON" as parameter.

`__LL_ADC_IS_ENABLED_ALL_COMMON_INSTANCE`

Description:

- Helper macro to check if all ADC instances sharing the same ADC common instance are disabled.

Parameters:

- `__ADCXY_COMMON__`: ADC common instance (can be set directly from CMSIS definition or by using helper macro)

Return value:

- Value: "0" if all ADC instances sharing the same ADC common instance are disabled. Value "1" if at least one ADC instance sharing the same ADC common instance is enabled.

Notes:

- This check is required by functions with setting conditioned to ADC state: All ADC instances of the ADC common group must be disabled. Refer to functions having argument `"ADCxy_COMMON"` as parameter. On devices with only 1 ADC common instance, parameter of this macro is useless and can be ignored (parameter kept for compatibility with devices featuring several ADC common instances).

[_LL_ADC_DIGITAL_SCALE](#)**Description:**

- Helper macro to define the ADC conversion data full-scale digital value corresponding to the selected ADC resolution.

Parameters:

- `__ADC_RESOLUTION__`: This parameter can be one of the following values:
 - `LL_ADC_RESOLUTION_12B`
 - `LL_ADC_RESOLUTION_10B`
 - `LL_ADC_RESOLUTION_8B`
 - `LL_ADC_RESOLUTION_6B`

Return value:

- ADC: conversion data equivalent voltage value (unit: mVolt)

Notes:

- ADC conversion data full-scale corresponds to voltage range determined by analog voltage references Vref+ and Vref- (refer to reference manual).

[_LL_ADC_CONVERT_DATA_RESOLUTION](#)**Description:**

- Helper macro to convert the ADC conversion data from a resolution to another resolution.

Parameters:

- `__DATA__`: ADC conversion data to be converted
- `__ADC_RESOLUTION_CURRENT__`: Resolution of to the data to be converted This

parameter can be one of the following values:

- LL_ADC_RESOLUTION_12B
- LL_ADC_RESOLUTION_10B
- LL_ADC_RESOLUTION_8B
- LL_ADC_RESOLUTION_6B

- __ADC_RESOLUTION_TARGET__:

Resolution of the data after conversion This parameter can be one of the following values:

- LL_ADC_RESOLUTION_12B
- LL_ADC_RESOLUTION_10B
- LL_ADC_RESOLUTION_8B
- LL_ADC_RESOLUTION_6B

Return value:

- ADC: conversion data to the requested resolution

__LL_ADC_CALC_DATA_TO_VOLTAGE

Description:

- Helper macro to calculate the voltage (unit: mVolt) corresponding to a ADC conversion data (unit: digital value).

Parameters:

- __VREFANALOG_VOLTAGE__: Analog reference voltage (unit: mV)
- __ADC_DATA__: ADC conversion data (resolution 12 bits) (unit: digital value).
- __ADC_RESOLUTION__: This parameter can be one of the following values:
 - LL_ADC_RESOLUTION_12B
 - LL_ADC_RESOLUTION_10B
 - LL_ADC_RESOLUTION_8B
 - LL_ADC_RESOLUTION_6B

Return value:

- ADC: conversion data equivalent voltage value (unit: mVolt)

Notes:

- Analog reference voltage (V_{ref+}) must be either known from user board environment or can be calculated using ADC measurement and ADC helper macro __LL_ADC_CALC_VREFANALOG_VOLTAGE().

__LL_ADC_CALC_VREFANALOG_VOLTAGE

Description:

- Helper macro to calculate analog reference voltage (V_{ref+}) (unit: mVolt) from ADC conversion data of internal voltage reference V_{refInt} .

Parameters:

- __VREFINT_ADC_DATA__: ADC conversion

data (resolution 12 bits) of internal voltage reference VrefInt (unit: digital value).

- `__ADC_RESOLUTION__`: This parameter can be one of the following values:
 - `LL_ADC_RESOLUTION_12B`
 - `LL_ADC_RESOLUTION_10B`
 - `LL_ADC_RESOLUTION_8B`
 - `LL_ADC_RESOLUTION_6B`

Return value:

- Analog: reference voltage (unit: mV)

Notes:

- Computation is using VrefInt calibration value stored in system memory for each device during production. This voltage depends on user board environment: voltage level connected to pin Vref+. On devices with small package, the pin Vref+ is not present and internally bonded to pin Vdda. On this STM32 serie, calibration data of internal voltage reference VrefInt corresponds to a resolution of 12 bits, this is the recommended ADC resolution to convert voltage of internal voltage reference VrefInt. Otherwise, this macro performs the processing to scale ADC conversion data to 12 bits.

`LL_ADC_CALC_TEMPERATUR`

E

Description:

- Helper macro to calculate the temperature (unit: degree Celsius) from ADC conversion data of internal temperature sensor.

Parameters:

- `__VREFANALOG_VOLTAGE__`: Analog reference voltage (unit: mV)
- `__TEMPSENSOR_ADC_DATA__`: ADC conversion data of internal temperature sensor (unit: digital value).
- `__ADC_RESOLUTION__`: ADC resolution at which internal temperature sensor voltage has been measured. This parameter can be one of the following values:
 - `LL_ADC_RESOLUTION_12B`
 - `LL_ADC_RESOLUTION_10B`
 - `LL_ADC_RESOLUTION_8B`
 - `LL_ADC_RESOLUTION_6B`

Return value:

- Temperature: (unit: degree Celsius)

Notes:

- Computation is using temperature sensor calibration values stored in system memory for

each device during production. Calculation formula: Temperature = ((TS_ADC_DATA - TS_CAL1) * (TS_CAL2_TEMP - TS_CAL1_TEMP)) / (TS_CAL2 - TS_CAL1) + TS_CAL1_TEMP with TS_ADC_DATA = temperature sensor raw data measured by ADC Avg_Slope = (TS_CAL2 - TS_CAL1) / (TS_CAL2_TEMP - TS_CAL1_TEMP) TS_CAL1 = equivalent TS_ADC_DATA at temperature TEMP_DEGC_CAL1 (calibrated in factory) TS_CAL2 = equivalent TS_ADC_DATA at temperature TEMP_DEGC_CAL2 (calibrated in factory) Caution: Calculation relevancy under reserve that calibration parameters are correct (address and data). To calculate temperature using temperature sensor datasheet typical values (generic values less, therefore less accurate than calibrated values), use helper macro

`__LL_ADC_CALC_TEMPERATURE_TYP_PARAMETERS()`. As calculation input, the analog reference voltage (Vref+) must be defined as it impacts the ADC LSB equivalent voltage. Analog reference voltage (Vref+) must be either known from user board environment or can be calculated using ADC measurement and ADC helper macro

`__LL_ADC_CALC_VREFANALOG_VOLTAGE()`. On this STM32 serie, calibration data of temperature sensor corresponds to a resolution of 12 bits, this is the recommended ADC resolution to convert voltage of temperature sensor. Otherwise, this macro performs the processing to scale ADC conversion data to 12 bits.

`__LL_ADC_CALC_TEMPERATURE_TYP_PARAMS`

Description:

- Helper macro to calculate the temperature (unit: degree Celsius) from ADC conversion data of internal temperature sensor.

Parameters:

- `__TEMPSENSOR_TYP_AVGSLOPE__`: Device datasheet data: Temperature sensor slope typical value (unit: uV/DegCelsius). On STM32L1, refer to device datasheet parameter "Avg_Slope".
- `__TEMPSENSOR_TYP_CALX_V__`: Device datasheet data: Temperature sensor voltage typical value (at temperature and Vref+ defined in parameters below) (unit: mV). On STM32L1, refer to device datasheet parameter "V110" (corresponding to TS_CAL2).
- `__TEMPSENSOR_CALX_TEMP__`: Device datasheet data: Temperature at which

temperature sensor voltage (see parameter above) is corresponding (unit: mV)

- `__VREFANALOG_VOLTAGE__`: Analog voltage reference (Vref+) voltage (unit: mV)
- `__TEMPSENSOR_ADC_DATA__`: ADC conversion data of internal temperature sensor (unit: digital value).
- `__ADC_RESOLUTION__`: ADC resolution at which internal temperature sensor voltage has been measured. This parameter can be one of the following values:
 - `LL_ADC_RESOLUTION_12B`
 - `LL_ADC_RESOLUTION_10B`
 - `LL_ADC_RESOLUTION_8B`
 - `LL_ADC_RESOLUTION_6B`

Return value:

- Temperature: (unit: degree Celsius)

Notes:

- Computation is using temperature sensor typical values (refer to device datasheet). Calculation formula: Temperature = $(TS_TYP_CALx_VOLT(\mu V) - TS_ADC_DATA * Conversion_{\mu V}) / Avg_Slope + CALx_TEMP$ with TS_ADC_DATA = temperature sensor raw data measured by ADC (unit: digital value) Avg_Slope = temperature sensor slope (unit: $\mu V/Degree\ Celsius$) $TS_TYP_CALx_VOLT$ = temperature sensor digital value at temperature $CALx_TEMP$ (unit: mV) Caution: Calculation relevancy under reserve the temperature sensor of the current device has characteristics in line with datasheet typical values. If temperature sensor calibration values are available on on this device (presence of macro `__LL_ADC_CALC_TEMPERATURE()`), temperature calculation will be more accurate using helper macro `__LL_ADC_CALC_TEMPERATURE()`. As calculation input, the analog reference voltage (Vref+) must be defined as it impacts the ADC LSB equivalent voltage. Analog reference voltage (Vref+) must be either known from user board environment or can be calculated using ADC measurement and ADC helper macro `__LL_ADC_CALC_VREFANALOG_VOLTAGE()`. ADC measurement data must correspond to a resolution of 12bits (full scale digital value 4095). If not the case, the data must be preliminarily rescaled to an equivalent resolution of 12 bits.

Common write and read registers Macros



LL_ADC_WriteReg**Description:**

- Write a value in ADC register.

Parameters:

- __INSTANCE__: ADC Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_ADC_ReadReg**Description:**

- Read a value in ADC register.

Parameters:

- __INSTANCE__: ADC Instance
- __REG__: Register to be read

Return value:

- Register: value

49 LL BUS Generic Driver

49.1 BUS Firmware driver API description

49.1.1 Detailed description of functions

LL_AHB1_GRP1_EnableClock

Function name **__STATIC_INLINE void LL_AHB1_GRP1_EnableClock(
 uint32_t Periph)**

Function description Enable AHB1 peripherals clock.

- Parameters
- **Periph:** This parameter can be a combination of the following values: (*) value not defined in all devices.
 - LL_AHB1_GRP1_PERIPH_GPIOA
 - LL_AHB1_GRP1_PERIPH_GPIOB
 - LL_AHB1_GRP1_PERIPH_GPIOC
 - LL_AHB1_GRP1_PERIPH_GPIOD
 - LL_AHB1_GRP1_PERIPH_GPIOE (*)
 - LL_AHB1_GRP1_PERIPH_GPIOH
 - LL_AHB1_GRP1_PERIPH_GPIOF (*)
 - LL_AHB1_GRP1_PERIPH_GPIOG (*)
 - LL_AHB1_GRP1_PERIPH_CRC
 - LL_AHB1_GRP1_PERIPH_FLASH
 - LL_AHB1_GRP1_PERIPH_DMA1
 - LL_AHB1_GRP1_PERIPH_DMA2 (*)
 - LL_AHB1_GRP1_PERIPH_CRYP (*)
 - LL_AHB1_GRP1_PERIPH_FSMC (*)

Return values

- Reference Manual to LL API cross reference:
- AHBENR GPIOAEN LL_AHB1_GRP1_EnableClock
 - AHBENR GPIOBEN LL_AHB1_GRP1_EnableClock
 - AHBENR GPIOCEN LL_AHB1_GRP1_EnableClock
 - AHBENR GPIODEN LL_AHB1_GRP1_EnableClock
 - AHBENR GPIOEEN LL_AHB1_GRP1_EnableClock
 - AHBENR GPIOHEN LL_AHB1_GRP1_EnableClock
 - AHBENR GPIOFEN LL_AHB1_GRP1_EnableClock
 - AHBENR GPIOGEN LL_AHB1_GRP1_EnableClock
 - AHBENR CRCEN LL_AHB1_GRP1_EnableClock
 - AHBENR FLITFEN LL_AHB1_GRP1_EnableClock
 - AHBENR DMA1EN LL_AHB1_GRP1_EnableClock
 - AHBENR DMA2EN LL_AHB1_GRP1_EnableClock
 - AHBENR AESEN LL_AHB1_GRP1_EnableClock
 - AHBENR FSMCEN LL_AHB1_GRP1_EnableClock

LL_AHB1_GRP1_IsEnabledClock

Function name **__STATIC_INLINE uint32_t LL_AHB1_GRP1_IsEnabledClock(
 uint32_t Periph)**

Function description Check if AHB1 peripheral clock is enabled or not.

- | | |
|---|---|
| Parameters | <ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_AHB1_GRP1_PERIPH_GPIOA - LL_AHB1_GRP1_PERIPH_GPIOB - LL_AHB1_GRP1_PERIPH_GPIOC - LL_AHB1_GRP1_PERIPH_GPIOD - LL_AHB1_GRP1_PERIPH_GPIOE (*) - LL_AHB1_GRP1_PERIPH_GPIOH - LL_AHB1_GRP1_PERIPH_GPIOF (*) - LL_AHB1_GRP1_PERIPH_GPIOG (*) - LL_AHB1_GRP1_PERIPH_CRC - LL_AHB1_GRP1_PERIPH_FLASH - LL_AHB1_GRP1_PERIPH_DMA1 - LL_AHB1_GRP1_PERIPH_DMA2 (*) - LL_AHB1_GRP1_PERIPH_CRYP (*) - LL_AHB1_GRP1_PERIPH_FSMC (*) |
| Return values | <ul style="list-style-type: none"> • State: of Periph (1 or 0). |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> • AHBENR GPIOAEN LL_AHB1_GRP1_IsEnabledClock • AHBENR GPIOBEN LL_AHB1_GRP1_IsEnabledClock • AHBENR GPIOCEN LL_AHB1_GRP1_IsEnabledClock • AHBENR GPIODEN LL_AHB1_GRP1_IsEnabledClock • AHBENR GPIOEEN LL_AHB1_GRP1_IsEnabledClock • AHBENR GPIOHEN LL_AHB1_GRP1_IsEnabledClock • AHBENR GPIOFEN LL_AHB1_GRP1_IsEnabledClock • AHBENR GPIOGEN LL_AHB1_GRP1_IsEnabledClock • AHBENR CRCEN LL_AHB1_GRP1_IsEnabledClock • AHBENR FLITFEN LL_AHB1_GRP1_IsEnabledClock • AHBENR DMA1EN LL_AHB1_GRP1_IsEnabledClock • AHBENR DMA2EN LL_AHB1_GRP1_IsEnabledClock • AHBENR AESEN LL_AHB1_GRP1_IsEnabledClock • AHBENR FSMCEN LL_AHB1_GRP1_IsEnabledClock |

LL_AHB1_GRP1_DisableClock

- | | |
|----------------------|---|
| Function name | <code>__STATIC_INLINE void LL_AHB1_GRP1_DisableClock
(uint32_t Periph)</code> |
| Function description | Disable AHB1 peripherals clock. |
| Parameters | <ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_AHB1_GRP1_PERIPH_GPIOA - LL_AHB1_GRP1_PERIPH_GPIOB - LL_AHB1_GRP1_PERIPH_GPIOC - LL_AHB1_GRP1_PERIPH_GPIOD - LL_AHB1_GRP1_PERIPH_GPIOE (*) - LL_AHB1_GRP1_PERIPH_GPIOH - LL_AHB1_GRP1_PERIPH_GPIOF (*) - LL_AHB1_GRP1_PERIPH_GPIOG (*) - LL_AHB1_GRP1_PERIPH_CRC - LL_AHB1_GRP1_PERIPH_FLASH - LL_AHB1_GRP1_PERIPH_DMA1 - LL_AHB1_GRP1_PERIPH_DMA2 (*) |

Return values Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> - LL_AHB1_GRP1_PERIPH_CRYP (*) - LL_AHB1_GRP1_PERIPH_FSMC (*)
<ul style="list-style-type: none"> • None: 	<ul style="list-style-type: none"> • AHBENR GPIOAEN LL_AHB1_GRP1_DisableClock • AHBENR GPIOBEN LL_AHB1_GRP1_DisableClock • AHBENR GPIOCEN LL_AHB1_GRP1_DisableClock • AHBENR GPIODEN LL_AHB1_GRP1_DisableClock • AHBENR GPIOEEN LL_AHB1_GRP1_DisableClock • AHBENR GPIOHEN LL_AHB1_GRP1_DisableClock • AHBENR GPIOFEN LL_AHB1_GRP1_DisableClock • AHBENR GPIOGEN LL_AHB1_GRP1_DisableClock • AHBENR CRCEN LL_AHB1_GRP1_DisableClock • AHBENR FLTFEN LL_AHB1_GRP1_DisableClock • AHBENR DMA1EN LL_AHB1_GRP1_DisableClock • AHBENR DMA2EN LL_AHB1_GRP1_DisableClock • AHBENR AESEN LL_AHB1_GRP1_DisableClock • AHBENR FSMCEN LL_AHB1_GRP1_DisableClock

LL_AHB1_GRP1_ForceReset

Function name Function description Parameters	<code>__STATIC_INLINE void LL_AHB1_GRP1_ForceReset (uint32_t Periph)</code> Force AHB1 peripherals reset. <ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices.
<ul style="list-style-type: none"> • None: 	<ul style="list-style-type: none"> - LL_AHB1_GRP1_PERIPH_ALL - LL_AHB1_GRP1_PERIPH_GPIOA - LL_AHB1_GRP1_PERIPH_GPIOB - LL_AHB1_GRP1_PERIPH_GPIOC - LL_AHB1_GRP1_PERIPH_GPIOD - LL_AHB1_GRP1_PERIPH_GPIOE (*) - LL_AHB1_GRP1_PERIPH_GPIOH - LL_AHB1_GRP1_PERIPH_GPIOF (*) - LL_AHB1_GRP1_PERIPH_GPIOG (*) - LL_AHB1_GRP1_PERIPH_CRC - LL_AHB1_GRP1_PERIPH_FLASH - LL_AHB1_GRP1_PERIPH_DMA1 - LL_AHB1_GRP1_PERIPH_DMA2 (*) - LL_AHB1_GRP1_PERIPH_CRYP (*) - LL_AHB1_GRP1_PERIPH_FSMC (*)

- AHBRSTR FLITFRST LL_AHB1_GRP1_ForceReset
- AHBRSTR DMA1RST LL_AHB1_GRP1_ForceReset
- AHBRSTR DMA2RST LL_AHB1_GRP1_ForceReset
- AHBRSTR AESRST LL_AHB1_GRP1_ForceReset
- AHBRSTR FSMCRST LL_AHB1_GRP1_ForceReset

LL_AHB1_GRP1_ReleaseReset

Function name **_STATIC_INLINE void LL_AHB1_GRP1_ReleaseReset (uint32_t Periph)**

Function description Release AHB1 peripherals reset.

- Parameters
- **Periph:** This parameter can be a combination of the following values: (*) value not defined in all devices.
 - LL_AHB1_GRP1_PERIPH_ALL
 - LL_AHB1_GRP1_PERIPH_GPIOA
 - LL_AHB1_GRP1_PERIPH_GPIOB
 - LL_AHB1_GRP1_PERIPH_GPIOC
 - LL_AHB1_GRP1_PERIPH_GPIOD
 - LL_AHB1_GRP1_PERIPH_GPIOE (*)
 - LL_AHB1_GRP1_PERIPH_GPIOH
 - LL_AHB1_GRP1_PERIPH_GPIOF (*)
 - LL_AHB1_GRP1_PERIPH_GPIOG (*)
 - LL_AHB1_GRP1_PERIPH_CRC
 - LL_AHB1_GRP1_PERIPH_FLASH
 - LL_AHB1_GRP1_PERIPH_DMA1
 - LL_AHB1_GRP1_PERIPH_DMA2 (*)
 - LL_AHB1_GRP1_PERIPH_CRYP (*)
 - LL_AHB1_GRP1_PERIPH_FSMC (*)

- Return values
- **None:**

- Reference Manual to
LL API cross
reference:
- AHBRSTR GPIOARST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR GPIOBRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR GPIOCRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR GPIODRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR GPIOERST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR GPIOHRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR GPIOFRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR GPIOGRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR CRCRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR FLITFRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR DMA1RST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR DMA2RST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR AESRST LL_AHB1_GRP1_ReleaseReset
 - AHBRSTR FSMCRST LL_AHB1_GRP1_ReleaseReset

LL_AHB1_GRP1_EnableClockSleep

Function name **_STATIC_INLINE void LL_AHB1_GRP1_EnableClockSleep (uint32_t Periph)**

Function description Enable AHB1 peripherals clock during Low Power (Sleep) mode.

Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_AHB1_GRP1_PERIPH_GPIOA - LL_AHB1_GRP1_PERIPH_GPIOB - LL_AHB1_GRP1_PERIPH_GPIOC - LL_AHB1_GRP1_PERIPH_GPIOD - LL_AHB1_GRP1_PERIPH_GPIOE (*) - LL_AHB1_GRP1_PERIPH_GPIOH - LL_AHB1_GRP1_PERIPH_GPIOF (*) - LL_AHB1_GRP1_PERIPH_GPIOG (*) - LL_AHB1_GRP1_PERIPH_CRC - LL_AHB1_GRP1_PERIPH_FLASH - LL_AHB1_GRP1_PERIPH_SRAM - LL_AHB1_GRP1_PERIPH_DMA1 - LL_AHB1_GRP1_PERIPH_DMA2 (*) - LL_AHB1_GRP1_PERIPH_CRYP (*) - LL_AHB1_GRP1_PERIPH_FSMC (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • AHBLPENR GPIOALPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR GPIOBLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR GPIOCLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR GPIODLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR GPIOELPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR GPIOFLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR GPIOGLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR CRCLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR FLITFLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR SRAMLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR DMA1LPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR DMA2LPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR AESLPEN LL_AHB1_GRP1_EnableClockSleep • AHBLPENR FSMCLPEN LL_AHB1_GRP1_EnableClockSleep

LL_AHB1_GRP1_DisableClockSleep

Function name

```
_STATIC_INLINE void LL_AHB1_GRP1_DisableClockSleep  
(uint32_t Periph)
```

Function description	Disable AHB1 peripherals clock during Low Power (Sleep) mode.
Parameters	<ul style="list-style-type: none"> Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_AHB1_GRP1_PERIPH_GPIOA - LL_AHB1_GRP1_PERIPH_GPIOB - LL_AHB1_GRP1_PERIPH_GPIOC - LL_AHB1_GRP1_PERIPH_GPIOD - LL_AHB1_GRP1_PERIPH_GPIOE (*) - LL_AHB1_GRP1_PERIPH_GPIOH - LL_AHB1_GRP1_PERIPH_GPIOF (*) - LL_AHB1_GRP1_PERIPH_GPIOG (*) - LL_AHB1_GRP1_PERIPH_CRC - LL_AHB1_GRP1_PERIPH_FLASH - LL_AHB1_GRP1_PERIPH_SRAM - LL_AHB1_GRP1_PERIPH_DMA1 - LL_AHB1_GRP1_PERIPH_DMA2 (*) - LL_AHB1_GRP1_PERIPH_CRYP (*) - LL_AHB1_GRP1_PERIPH_FSMC (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • AHBLPENR GPIOALPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR GPIOBLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR GPIOCLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR GPIOODLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR GPIOELPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR GPIOOHPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR GPIOFLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR GPIOGLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR CRCLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR FLITFLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR SRAMLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR DMA1LPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR DMA2LPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR AESLPEN LL_AHB1_GRP1_DisableClockSleep • AHBLPENR FSMCLPEN LL_AHB1_GRP1_DisableClockSleep

LL_APB1_GRP1_EnableClock

Function name	<code>_STATIC_INLINE void LL_APB1_GRP1_EnableClock (uint32_t Periph)</code>
Function description	Enable APB1 peripherals clock.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - <code>LL_APB1_GRP1_PERIPH_TIM2</code> - <code>LL_APB1_GRP1_PERIPH_TIM3</code> - <code>LL_APB1_GRP1_PERIPH_TIM4</code> - <code>LL_APB1_GRP1_PERIPH_TIM5 (*)</code> - <code>LL_APB1_GRP1_PERIPH_TIM6</code> - <code>LL_APB1_GRP1_PERIPH_TIM7</code> - <code>LL_APB1_GRP1_PERIPH_LCD (*)</code> - <code>LL_APB1_GRP1_PERIPH_WWDG</code> - <code>LL_APB1_GRP1_PERIPH_SPI2</code> - <code>LL_APB1_GRP1_PERIPH_SPI3 (*)</code> - <code>LL_APB1_GRP1_PERIPH_USART2</code> - <code>LL_APB1_GRP1_PERIPH_USART3</code> - <code>LL_APB1_GRP1_PERIPH_UART4 (*)</code> - <code>LL_APB1_GRP1_PERIPH_UART5 (*)</code> - <code>LL_APB1_GRP1_PERIPH_I2C1</code> - <code>LL_APB1_GRP1_PERIPH_I2C2</code> - <code>LL_APB1_GRP1_PERIPH_USB</code> - <code>LL_APB1_GRP1_PERIPH_PWR</code> - <code>LL_APB1_GRP1_PERIPH_DAC1</code> - <code>LL_APB1_GRP1_PERIPH_COMP</code> - <code>LL_APB1_GRP1_PERIPH_OPAMP (*)</code>
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • APB1ENR TIM2EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR TIM3EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR TIM4EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR TIM5EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR TIM6EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR TIM7EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR LCDEN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR WWDGEN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR SPI2EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR SPI3EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR USART2EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR USART3EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR UART4EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR UART5EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR I2C1EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR I2C2EN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR USBEN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR PWREN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR DACEN <code>LL_APB1_GRP1_EnableClock</code> • APB1ENR COMPEN <code>LL_APB1_GRP1_EnableClock</code>

LL_APB1_GRP1_IsEnabledClock

Function name **`_STATIC_INLINE uint32_t LL_APB1_GRP1_IsEnabledClock
(uint32_t Periph)`**

Function description Check if APB1 peripheral clock is enabled or not.

- Parameters
- **Periph:** This parameter can be a combination of the following values: (*) value not defined in all devices.
 - `LL_APB1_GRP1_PERIPH_TIM2`
 - `LL_APB1_GRP1_PERIPH_TIM3`
 - `LL_APB1_GRP1_PERIPH_TIM4`
 - `LL_APB1_GRP1_PERIPH_TIM5 (*)`
 - `LL_APB1_GRP1_PERIPH_TIM6`
 - `LL_APB1_GRP1_PERIPH_TIM7`
 - `LL_APB1_GRP1_PERIPH_LCD (*)`
 - `LL_APB1_GRP1_PERIPH_WWDG`
 - `LL_APB1_GRP1_PERIPH_SPI2`
 - `LL_APB1_GRP1_PERIPH_SPI3 (*)`
 - `LL_APB1_GRP1_PERIPH_USART2`
 - `LL_APB1_GRP1_PERIPH_USART3`
 - `LL_APB1_GRP1_PERIPH_UART4 (*)`
 - `LL_APB1_GRP1_PERIPH_UART5 (*)`
 - `LL_APB1_GRP1_PERIPH_I2C1`
 - `LL_APB1_GRP1_PERIPH_I2C2`
 - `LL_APB1_GRP1_PERIPH_USB`
 - `LL_APB1_GRP1_PERIPH_PWR`
 - `LL_APB1_GRP1_PERIPH_DAC1`
 - `LL_APB1_GRP1_PERIPH_COMP`
 - `LL_APB1_GRP1_PERIPH_OPAMP (*)`

- Return values
- **State:** of Periph (1 or 0).

- Reference Manual to
LL API cross
reference:
- `APB1ENR TIM2EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR TIM3EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR TIM4EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR TIM5EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR TIM6EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR TIM7EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR LCDEN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR WWDGEN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR SPI2EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR SPI3EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR USART2EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR USART3EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR UART4EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR UART5EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR I2C1EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR I2C2EN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR USBEN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR PWREN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR DACEN LL_APB1_GRP1_IsEnabledClock`
 - `APB1ENR COMPEN LL_APB1_GRP1_IsEnabledClock`

LL_APB1_GRP1_DisableClock

Function name **`_STATIC_INLINE void LL_APB1_GRP1_DisableClock
(uint32_t Periph)`**

Function description Disable APB1 peripherals clock.

- Parameters
- **Periph:** This parameter can be a combination of the following values: (*) value not defined in all devices.
 - `LL_APB1_GRP1_PERIPH_TIM2`
 - `LL_APB1_GRP1_PERIPH_TIM3`
 - `LL_APB1_GRP1_PERIPH_TIM4`
 - `LL_APB1_GRP1_PERIPH_TIM5 (*)`
 - `LL_APB1_GRP1_PERIPH_TIM6`
 - `LL_APB1_GRP1_PERIPH_TIM7`
 - `LL_APB1_GRP1_PERIPH_LCD (*)`
 - `LL_APB1_GRP1_PERIPH_WWDG`
 - `LL_APB1_GRP1_PERIPH_SPI2`
 - `LL_APB1_GRP1_PERIPH_SPI3 (*)`
 - `LL_APB1_GRP1_PERIPH_USART2`
 - `LL_APB1_GRP1_PERIPH_USART3`
 - `LL_APB1_GRP1_PERIPH_UART4 (*)`
 - `LL_APB1_GRP1_PERIPH_UART5 (*)`
 - `LL_APB1_GRP1_PERIPH_I2C1`
 - `LL_APB1_GRP1_PERIPH_I2C2`
 - `LL_APB1_GRP1_PERIPH_USB`
 - `LL_APB1_GRP1_PERIPH_PWR`
 - `LL_APB1_GRP1_PERIPH_DAC1`
 - `LL_APB1_GRP1_PERIPH_COMP`
 - `LL_APB1_GRP1_PERIPH_OPAMP (*)`

- Return values
- **None:**

- Reference Manual to LL API cross reference:
- `APB1ENR TIM2EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR TIM3EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR TIM4EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR TIM5EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR TIM6EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR TIM7EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR LCDEN LL_APB1_GRP1_DisableClock`
 - `APB1ENR WWDGEN LL_APB1_GRP1_DisableClock`
 - `APB1ENR SPI2EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR SPI3EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR USART2EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR USART3EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR UART4EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR UART5EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR I2C1EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR I2C2EN LL_APB1_GRP1_DisableClock`
 - `APB1ENR USBEN LL_APB1_GRP1_DisableClock`
 - `APB1ENR PWREN LL_APB1_GRP1_DisableClock`
 - `APB1ENR DACEN LL_APB1_GRP1_DisableClock`
 - `APB1ENR COMPEN LL_APB1_GRP1_DisableClock`

LL_APB1_GRP1_ForceReset

Function name	<code>__STATIC_INLINE void LL_APB1_GRP1_ForceReset (uint32_t Periph)</code>
Function description	Force APB1 peripherals reset.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - <code>LL_APB1_GRP1_PERIPH_ALL</code> - <code>LL_APB1_GRP1_PERIPH_TIM2</code> - <code>LL_APB1_GRP1_PERIPH_TIM3</code> - <code>LL_APB1_GRP1_PERIPH_TIM4</code> - <code>LL_APB1_GRP1_PERIPH_TIM5 (*)</code> - <code>LL_APB1_GRP1_PERIPH_TIM6</code> - <code>LL_APB1_GRP1_PERIPH_TIM7</code> - <code>LL_APB1_GRP1_PERIPH_LCD (*)</code> - <code>LL_APB1_GRP1_PERIPH_WWDG</code> - <code>LL_APB1_GRP1_PERIPH_SPI2</code> - <code>LL_APB1_GRP1_PERIPH_SPI3 (*)</code> - <code>LL_APB1_GRP1_PERIPH_USART2</code> - <code>LL_APB1_GRP1_PERIPH_USART3</code> - <code>LL_APB1_GRP1_PERIPH_UART4 (*)</code> - <code>LL_APB1_GRP1_PERIPH_UART5 (*)</code> - <code>LL_APB1_GRP1_PERIPH_I2C1</code> - <code>LL_APB1_GRP1_PERIPH_I2C2</code> - <code>LL_APB1_GRP1_PERIPH_USB</code> - <code>LL_APB1_GRP1_PERIPH_PWR</code> - <code>LL_APB1_GRP1_PERIPH_DAC1</code> - <code>LL_APB1_GRP1_PERIPH_COMP</code> - <code>LL_APB1_GRP1_PERIPH_OPAMP (*)</code>
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>APB1RSTR TIM2RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR TIM3RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR TIM4RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR TIM5RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR TIM6RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR TIM7RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR LCDRST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR WWDGRST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR SPI2RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR SPI3RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR USART2RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR USART3RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR UART4RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR UART5RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR I2C1RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR I2C2RST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR USBRST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR PWRRST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR DACRST LL_APB1_GRP1_ForceReset</code> • <code>APB1RSTR COMPRST LL_APB1_GRP1_ForceReset</code>

LL_APB1_GRP1_ReleaseReset

Function name	<code>__STATIC_INLINE void LL_APB1_GRP1_ReleaseReset (uint32_t Periph)</code>
Function description	Release APB1 peripherals reset.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - <code>LL_APB1_GRP1_PERIPH_ALL</code> - <code>LL_APB1_GRP1_PERIPH_TIM2</code> - <code>LL_APB1_GRP1_PERIPH_TIM3</code> - <code>LL_APB1_GRP1_PERIPH_TIM4</code> - <code>LL_APB1_GRP1_PERIPH_TIM5 (*)</code> - <code>LL_APB1_GRP1_PERIPH_TIM6</code> - <code>LL_APB1_GRP1_PERIPH_TIM7</code> - <code>LL_APB1_GRP1_PERIPH_LCD (*)</code> - <code>LL_APB1_GRP1_PERIPH_WWDG</code> - <code>LL_APB1_GRP1_PERIPH_SPI2</code> - <code>LL_APB1_GRP1_PERIPH_SPI3 (*)</code> - <code>LL_APB1_GRP1_PERIPH_USART2</code> - <code>LL_APB1_GRP1_PERIPH_USART3</code> - <code>LL_APB1_GRP1_PERIPH_UART4 (*)</code> - <code>LL_APB1_GRP1_PERIPH_UART5 (*)</code> - <code>LL_APB1_GRP1_PERIPH_I2C1</code> - <code>LL_APB1_GRP1_PERIPH_I2C2</code> - <code>LL_APB1_GRP1_PERIPH_USB</code> - <code>LL_APB1_GRP1_PERIPH_PWR</code> - <code>LL_APB1_GRP1_PERIPH_DAC1</code> - <code>LL_APB1_GRP1_PERIPH_COMP</code> - <code>LL_APB1_GRP1_PERIPH_OPAMP (*)</code>
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>APB1RSTR TIM2RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR TIM3RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR TIM4RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR TIM5RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR TIM6RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR TIM7RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR LCDRST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR WWDGRST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR SPI2RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR SPI3RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR USART2RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR USART3RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR UART4RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR UART5RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR I2C1RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR I2C2RST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR USBRST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR PWRrst LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR DACRST LL_APB1_GRP1_ReleaseReset</code> • <code>APB1RSTR COMPRST LL_APB1_GRP1_ReleaseReset</code>

LL_APB1_GRP1_EnableClockSleep

Function name **`_STATIC_INLINE void LL_APB1_GRP1_EnableClockSleep
(uint32_t Periph)`**

Function description Enable APB1 peripherals clock during Low Power (Sleep) mode.

Parameters • **Periph**: This parameter can be a combination of the following values: (*) value not defined in all devices.

- `LL_APB1_GRP1_PERIPH_TIM2`
- `LL_APB1_GRP1_PERIPH_TIM3`
- `LL_APB1_GRP1_PERIPH_TIM4`
- `LL_APB1_GRP1_PERIPH_TIM5 (*)`
- `LL_APB1_GRP1_PERIPH_TIM6`
- `LL_APB1_GRP1_PERIPH_TIM7`
- `LL_APB1_GRP1_PERIPH_LCD (*)`
- `LL_APB1_GRP1_PERIPH_WWDG`
- `LL_APB1_GRP1_PERIPH_SPI2`
- `LL_APB1_GRP1_PERIPH_SPI3 (*)`
- `LL_APB1_GRP1_PERIPH_USART2`
- `LL_APB1_GRP1_PERIPH_USART3`
- `LL_APB1_GRP1_PERIPH_UART4 (*)`
- `LL_APB1_GRP1_PERIPH_UART5 (*)`
- `LL_APB1_GRP1_PERIPH_I2C1`
- `LL_APB1_GRP1_PERIPH_I2C2`
- `LL_APB1_GRP1_PERIPH_USB`
- `LL_APB1_GRP1_PERIPH_PWR`
- `LL_APB1_GRP1_PERIPH_DAC1`
- `LL_APB1_GRP1_PERIPH_COMP`
- `LL_APB1_GRP1_PERIPH_OPAMP (*)`

Return values • **None**:

Reference Manual to LL API cross reference:

- APB1LPENR TIM2LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR TIM3LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR TIM4LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR TIM5LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR TIM6LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR TIM7LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR LCDLPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR WWDGLPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR SPI2LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR SPI3LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR USART2LPEN
`LL_APB1_GRP1_EnableClockSleep`
- APB1LPENR USART3LPEN
`LL_APB1_GRP1_EnableClockSleep`

- LL_APB1_GRP1_EnableClockSleep
- APB1LPENR UART4LPEN
- LL_APB1_GRP1_EnableClockSleep
- APB1LPENR UART5LPEN
- LL_APB1_GRP1_EnableClockSleep
- APB1LPENR I2C1LPEN
- LL_APB1_GRP1_EnableClockSleep
- APB1LPENR I2C2LPEN
- LL_APB1_GRP1_EnableClockSleep
- APB1LPENR USBLPEN
- LL_APB1_GRP1_EnableClockSleep
- APB1LPENR PWRLPEN
- LL_APB1_GRP1_EnableClockSleep
- APB1LPENR DACLPEN
- LL_APB1_GRP1_EnableClockSleep
- APB1LPENR COMPLPEN
- LL_APB1_GRP1_EnableClockSleep

LL_APB1_GRP1_DisableClockSleep

Function name	<code>__STATIC_INLINE void LL_APB1_GRP1_DisableClockSleep (uint32_t Periph)</code>
Function description	Disable APB1 peripherals clock during Low Power (Sleep) mode.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_APB1_GRP1_PERIPH_TIM2 - LL_APB1_GRP1_PERIPH_TIM3 - LL_APB1_GRP1_PERIPH_TIM4 - LL_APB1_GRP1_PERIPH_TIM5 (*) - LL_APB1_GRP1_PERIPH_TIM6 - LL_APB1_GRP1_PERIPH_TIM7 - LL_APB1_GRP1_PERIPH_LCD (*) - LL_APB1_GRP1_PERIPH_WWDG - LL_APB1_GRP1_PERIPH_SPI2 - LL_APB1_GRP1_PERIPH_SPI3 (*) - LL_APB1_GRP1_PERIPH_USART2 - LL_APB1_GRP1_PERIPH_USART3 - LL_APB1_GRP1_PERIPH_UART4 (*) - LL_APB1_GRP1_PERIPH_UART5 (*) - LL_APB1_GRP1_PERIPH_I2C1 - LL_APB1_GRP1_PERIPH_I2C2 - LL_APB1_GRP1_PERIPH_USB - LL_APB1_GRP1_PERIPH_PWR - LL_APB1_GRP1_PERIPH_DAC1 - LL_APB1_GRP1_PERIPH_COMP - LL_APB1_GRP1_PERIPH_OPAMP (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • APB1LPENR TIM2LPEN • LL_APB1_GRP1_DisableClockSleep • APB1LPENR TIM3LPEN • LL_APB1_GRP1_DisableClockSleep

- APB1LPENR TIM4LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR TIM5LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR TIM6LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR TIM7LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR LCDLPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR WWDGLPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR SPI2LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR SPI3LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR USART2LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR USART3LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR UART4LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR UART5LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR I2C1LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR I2C2LPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR USBLPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR PWRLPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR DACLPEN
LL_APB1_GRP1_DisableClockSleep
- APB1LPENR COMPLPEN
LL_APB1_GRP1_DisableClockSleep

LL_APB2_GRP1_EnableClock

Function name	<code>STATIC_INLINE void LL_APB2_GRP1_EnableClock (uint32_t Periph)</code>
Function description	Enable APB2 peripherals clock.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_APB2_GRP1_PERIPH_SYSCFG - LL_APB2_GRP1_PERIPH_TIM9 - LL_APB2_GRP1_PERIPH_TIM10 - LL_APB2_GRP1_PERIPH_TIM11 - LL_APB2_GRP1_PERIPH_ADC1 - LL_APB2_GRP1_PERIPH_SDIO (*) - LL_APB2_GRP1_PERIPH_SPI1 - LL_APB2_GRP1_PERIPH_USART1

Return values Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • None: • APB2ENR SYSCFGEN LL_APB2_GRP1_EnableClock • APB2ENR TIM9EN LL_APB2_GRP1_EnableClock • APB2ENR TIM10EN LL_APB2_GRP1_EnableClock • APB2ENR TIM11EN LL_APB2_GRP1_EnableClock • APB2ENR ADC1EN LL_APB2_GRP1_EnableClock • APB2ENR SDIOEN LL_APB2_GRP1_EnableClock • APB2ENR SPI1EN LL_APB2_GRP1_EnableClock • APB2ENR USART1EN LL_APB2_GRP1_EnableClock
--	--

LL_APB2_GRP1_IsEnabledClock

Function name Function description Parameters Return values Reference Manual to LL API cross reference:	<p>Function name <code>__STATIC_INLINE uint32_t LL_APB2_GRP1_IsEnabledClock(uint32_t Periph)</code></p> <p>Function description Check if APB2 peripheral clock is enabled or not.</p> <ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_APB2_GRP1_PERIPH_SYSCFG - LL_APB2_GRP1_PERIPH_TIM9 - LL_APB2_GRP1_PERIPH_TIM10 - LL_APB2_GRP1_PERIPH_TIM11 - LL_APB2_GRP1_PERIPH_ADC1 - LL_APB2_GRP1_PERIPH_SDIO (*) - LL_APB2_GRP1_PERIPH_SPI1 - LL_APB2_GRP1_PERIPH_USART1 • State: of Periph (1 or 0). <ul style="list-style-type: none"> • APB2ENR SYSCFGEN LL_APB2_GRP1_IsEnabledClock • APB2ENR TIM9EN LL_APB2_GRP1_IsEnabledClock • APB2ENR TIM10EN LL_APB2_GRP1_IsEnabledClock • APB2ENR TIM11EN LL_APB2_GRP1_IsEnabledClock • APB2ENR ADC1EN LL_APB2_GRP1_IsEnabledClock • APB2ENR SDIOEN LL_APB2_GRP1_IsEnabledClock • APB2ENR SPI1EN LL_APB2_GRP1_IsEnabledClock • APB2ENR USART1EN LL_APB2_GRP1_IsEnabledClock
---	---

LL_APB2_GRP1_DisableClock

Function name Function description Parameters	<p>Function name <code>__STATIC_INLINE void LL_APB2_GRP1_DisableClock(uint32_t Periph)</code></p> <p>Function description Disable APB2 peripherals clock.</p> <ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_APB2_GRP1_PERIPH_SYSCFG - LL_APB2_GRP1_PERIPH_TIM9 - LL_APB2_GRP1_PERIPH_TIM10 - LL_APB2_GRP1_PERIPH_TIM11 - LL_APB2_GRP1_PERIPH_ADC1 - LL_APB2_GRP1_PERIPH_SDIO (*) - LL_APB2_GRP1_PERIPH_SPI1
---	---

– LL_APB2_GRP1_PERIPH_USART1

Return values

**Reference Manual to
LL API cross
reference:**

- **None:**
- APB2ENR SYSCFGEN LL_APB2_GRP1_DisableClock
- APB2ENR TIM9EN LL_APB2_GRP1_DisableClock
- APB2ENR TIM10EN LL_APB2_GRP1_DisableClock
- APB2ENR TIM11EN LL_APB2_GRP1_DisableClock
- APB2ENR ADC1EN LL_APB2_GRP1_DisableClock
- APB2ENR SDIOEN LL_APB2_GRP1_DisableClock
- APB2ENR SPI1EN LL_APB2_GRP1_DisableClock
- APB2ENR USART1EN LL_APB2_GRP1_DisableClock

LL_APB2_GRP1_ForceReset

Function name

`_STATIC_INLINE void LL_APB2_GRP1_ForceReset (uint32_t Periphs)`

Function description

Force APB2 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values: (*) value not defined in all devices.
 - LL_APB2_GRP1_PERIPH_ALL
 - LL_APB2_GRP1_PERIPH_SYSCFG
 - LL_APB2_GRP1_PERIPH_TIM9
 - LL_APB2_GRP1_PERIPH_TIM10
 - LL_APB2_GRP1_PERIPH_TIM11
 - LL_APB2_GRP1_PERIPH_ADC1
 - LL_APB2_GRP1_PERIPH_SDIO (*)
 - LL_APB2_GRP1_PERIPH_SPI1
 - LL_APB2_GRP1_PERIPH_USART1

Return values

**Reference Manual to
LL API cross
reference:**

- **None:**
- APB2RSTR SYSCFGRST LL_APB2_GRP1_ForceReset
- APB2RSTR TIM9RST LL_APB2_GRP1_ForceReset
- APB2RSTR TIM10RST LL_APB2_GRP1_ForceReset
- APB2RSTR TIM11RST LL_APB2_GRP1_ForceReset
- APB2RSTR ADC1RST LL_APB2_GRP1_ForceReset
- APB2RSTR SDIORST LL_APB2_GRP1_ForceReset
- APB2RSTR SPI1RST LL_APB2_GRP1_ForceReset
- APB2RSTR USART1RST LL_APB2_GRP1_ForceReset

LL_APB2_GRP1_ReleaseReset

Function name

`_STATIC_INLINE void LL_APB2_GRP1_ReleaseReset (uint32_t Periphs)`

Function description

Release APB2 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values: (*) value not defined in all devices.
 - LL_APB2_GRP1_PERIPH_ALL
 - LL_APB2_GRP1_PERIPH_SYSCFG
 - LL_APB2_GRP1_PERIPH_TIM9
 - LL_APB2_GRP1_PERIPH_TIM10
 - LL_APB2_GRP1_PERIPH_TIM11

	<ul style="list-style-type: none"> - LL_APB2_GRP1_PERIPH_ADC1 - LL_APB2_GRP1_PERIPH_SDIO (*) - LL_APB2_GRP1_PERIPH_SPI1 - LL_APB2_GRP1_PERIPH_USART1
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • APB2RSTR SYSCFGRST LL_APB2_GRP1_ReleaseReset • APB2RSTR TIM9RST LL_APB2_GRP1_ReleaseReset • APB2RSTR TIM10RST LL_APB2_GRP1_ReleaseReset • APB2RSTR TIM11RST LL_APB2_GRP1_ReleaseReset • APB2RSTR ADC1RST LL_APB2_GRP1_ReleaseReset • APB2RSTR SDIORST LL_APB2_GRP1_ReleaseReset • APB2RSTR SPI1RST LL_APB2_GRP1_ReleaseReset • APB2RSTR USART1RST LL_APB2_GRP1_ReleaseReset

LL_APB2_GRP1_EnableClockSleep

Function name	<code>__STATIC_INLINE void LL_APB2_GRP1_EnableClockSleep(uint32_t Periph)</code>
Function description	Enable APB2 peripherals clock during Low Power (Sleep) mode.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_APB2_GRP1_PERIPH_SYSCFG - LL_APB2_GRP1_PERIPH_TIM9 - LL_APB2_GRP1_PERIPH_TIM10 - LL_APB2_GRP1_PERIPH_TIM11 - LL_APB2_GRP1_PERIPH_ADC1 - LL_APB2_GRP1_PERIPH_SDIO (*) - LL_APB2_GRP1_PERIPH_SPI1 - LL_APB2_GRP1_PERIPH_USART1
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • APB2LPENR SYSCFGLPEN LL_APB2_GRP1_EnableClockSleep • APB2LPENR TIM9LPEN LL_APB2_GRP1_EnableClockSleep • APB2LPENR TIM10LPEN LL_APB2_GRP1_EnableClockSleep • APB2LPENR TIM11LPEN LL_APB2_GRP1_EnableClockSleep • APB2LPENR ADC1LPEN LL_APB2_GRP1_EnableClockSleep • APB2LPENR SDIOLPEN LL_APB2_GRP1_EnableClockSleep • APB2LPENR SPI1LPEN LL_APB2_GRP1_EnableClockSleep • APB2LPENR USART1LPEN LL_APB2_GRP1_EnableClockSleep

LL_APB2_GRP1_DisableClockSleep

Function name	<code>__STATIC_INLINE void LL_APB2_GRP1_DisableClockSleep</code>
---------------	--

(uint32_t Periph)

Function description	Disable APB2 peripherals clock during Low Power (Sleep) mode.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_APB2_GRP1_PERIPH_SYSCFG - LL_APB2_GRP1_PERIPH_TIM9 - LL_APB2_GRP1_PERIPH_TIM10 - LL_APB2_GRP1_PERIPH_TIM11 - LL_APB2_GRP1_PERIPH_ADC1 - LL_APB2_GRP1_PERIPH_SDIO (*) - LL_APB2_GRP1_PERIPH_SPI1 - LL_APB2_GRP1_PERIPH_USART1
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • APB2LPENR SYSCFGLPEN LL_APB2_GRP1_DisableClockSleep • APB2LPENR TIM9LPEN LL_APB2_GRP1_DisableClockSleep • APB2LPENR TIM10LPEN LL_APB2_GRP1_DisableClockSleep • APB2LPENR TIM11LPEN LL_APB2_GRP1_DisableClockSleep • APB2LPENR ADC1LPEN LL_APB2_GRP1_DisableClockSleep • APB2LPENR SDIOLPEN LL_APB2_GRP1_DisableClockSleep • APB2LPENR SPI1LPEN LL_APB2_GRP1_DisableClockSleep • APB2LPENR USART1LPEN LL_APB2_GRP1_DisableClockSleep

49.2 BUS Firmware driver defines

49.2.1 BUS

AHB1 GRP1 PERIPH

LL_AHB1_GRP1_PERIPH_ALL
 LL_AHB1_GRP1_PERIPH_GPIOA
 LL_AHB1_GRP1_PERIPH_GPIOB
 LL_AHB1_GRP1_PERIPH_GPIOC
 LL_AHB1_GRP1_PERIPH_GPIOD
 LL_AHB1_GRP1_PERIPH_GPIOE
 LL_AHB1_GRP1_PERIPH_GPIOH
 LL_AHB1_GRP1_PERIPH_GPIOF
 LL_AHB1_GRP1_PERIPH_GPIOG
 LL_AHB1_GRP1_PERIPH_SRAM
 LL_AHB1_GRP1_PERIPH_CRC

LL_AHB1_GRP1_PERIPH_FLASH
LL_AHB1_GRP1_PERIPH_DMA1
LL_AHB1_GRP1_PERIPH_DMA2
LL_AHB1_GRP1_PERIPH_CRYP
LL_AHB1_GRP1_PERIPH_FSMC

APB1 GRP1 PERIPH

LL_APB1_GRP1_PERIPH_ALL
LL_APB1_GRP1_PERIPH_TIM2
LL_APB1_GRP1_PERIPH_TIM3
LL_APB1_GRP1_PERIPH_TIM4
LL_APB1_GRP1_PERIPH_TIM5
LL_APB1_GRP1_PERIPH_TIM6
LL_APB1_GRP1_PERIPH_TIM7
LL_APB1_GRP1_PERIPH_LCD
LL_APB1_GRP1_PERIPH_WWDG
LL_APB1_GRP1_PERIPH_SPI2
LL_APB1_GRP1_PERIPH_SPI3
LL_APB1_GRP1_PERIPH_USART2
LL_APB1_GRP1_PERIPH_USART3
LL_APB1_GRP1_PERIPH_UART4
LL_APB1_GRP1_PERIPH_UART5
LL_APB1_GRP1_PERIPH_I2C1
LL_APB1_GRP1_PERIPH_I2C2
LL_APB1_GRP1_PERIPH_USB
LL_APB1_GRP1_PERIPH_PWR
LL_APB1_GRP1_PERIPH_DAC1
LL_APB1_GRP1_PERIPH_COMP
LL_APB1_GRP1_PERIPH_OPAMP

APB2 GRP1 PERIPH

LL_APB2_GRP1_PERIPH_ALL
LL_APB2_GRP1_PERIPH_SYSCFG
LL_APB2_GRP1_PERIPH_TIM9
LL_APB2_GRP1_PERIPH_TIM10
LL_APB2_GRP1_PERIPH_TIM11
LL_APB2_GRP1_PERIPH_ADC1
LL_APB2_GRP1_PERIPH_SDIO

LL_APB2_GRP1_PERIPH_SPI1

LL_APB2_GRP1_PERIPH_USART1

50 LL COMP Generic Driver

50.1 COMP Firmware driver registers structures

50.1.1 LL_COMP_InitTypeDef

Data Fields

- *uint32_t PowerMode*
- *uint32_t InputPlus*
- *uint32_t InputMinus*
- *uint32_t OutputSelection*

Field Documentation

- ***uint32_t LL_COMP_InitTypeDef::PowerMode***
Set comparator operating mode to adjust power and speed. This parameter can be a value of **COMP_LL_EC_POWERMODE**This feature can be modified afterwards using unitary function **LL_COMP_SetPowerMode()**.
- ***uint32_t LL_COMP_InitTypeDef::InputPlus***
Set comparator input plus (non-inverting input). This parameter can be a value of **COMP_LL_EC_INPUT_PLUS**This feature can be modified afterwards using unitary function **LL_COMP_SetInputPlus()**.
- ***uint32_t LL_COMP_InitTypeDef::InputMinus***
Set comparator input minus (inverting input). This parameter can be a value of **COMP_LL_EC_INPUT_MINUS**This feature can be modified afterwards using unitary function **LL_COMP_SetInputMinus()**.
- ***uint32_t LL_COMP_InitTypeDef::OutputSelection***
Set comparator output selection. This parameter can be a value of **COMP_LL_EC_OUTPUT_SELECTION**This feature can be modified afterwards using unitary function **LL_COMP_SetOutputSelection()**.

50.2 COMP Firmware driver API description

50.2.1 Detailed description of functions

LL_COMP_SetCommonWindowMode

Function name	<code>STATIC_INLINE void LL_COMP_SetCommonWindowMode (COMP_Common_TypeDef * COMPxy_COMMON, uint32_t WindowMode)</code>
Function description	Set window mode of a pair of comparators instances (2 consecutive COMP instances odd and even COMP<x> and COMP<x+1>).
Parameters	<ul style="list-style-type: none"> • COMPxy_COMMON: Comparator common instance (can be set directly from CMSIS definition or by using helper macro <code>__LL_COMP_COMMON_INSTANCE()</code>) • WindowMode: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_COMP_WINDOWMODE_DISABLE</code> - <code>LL_COMP_WINDOWMODE_COMP2_INPUT_PLUS_COMMON</code>
Return values	<ul style="list-style-type: none"> • None:

- Reference Manual to LL API cross reference:
- CSR WNDWE LL_COMP_SetCommonWindowMode

LL_COMP_GetCommonWindowMode

Function name	<code>_STATIC_INLINE uint32_t LL_COMP_GetCommonWindowMode(COMP_Common_TypeDef * COMPxy_COMMON)</code>
Function description	Get window mode of a pair of comparators instances (2 consecutive COMP instances odd and even COMP<x> and COMP<x+1>).
Parameters	<ul style="list-style-type: none"> COMPxy_COMMON: Comparator common instance (can be set directly from CMSIS definition or by using helper macro <code>_LL_COMP_COMMON_INSTANCE()</code>)
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - <code>LL_COMP_WINDOWMODE_DISABLE</code> - <code>LL_COMP_WINDOWMODE_COMP2_INPUT_PLUS_COMMON</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR WNDWE LL_COMP_GetCommonWindowMode

LL_COMP_SetPowerMode

Function name	<code>_STATIC_INLINE void LL_COMP_SetPowerMode(COMP_TypeDef * COMPx, uint32_t PowerMode)</code>
Function description	Set comparator instance operating mode to adjust power and speed.
Parameters	<ul style="list-style-type: none"> COMPx: Comparator instance PowerMode: This parameter can be one of the following values: (1) Available only on COMP instance: COMP2. <ul style="list-style-type: none"> - <code>LL_COMP_POWERMODE_MEDIUMSPEED</code> (1) - <code>LL_COMP_POWERMODE_ULTRALOWPOWER</code> (1)
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> COMP2_CSR SPEED LL_COMP_SetPowerMode

LL_COMP_GetPowerMode

Function name	<code>_STATIC_INLINE uint32_t LL_COMP_GetPowerMode(COMP_TypeDef * COMPx)</code>
Function description	Get comparator instance operating mode to adjust power and speed.
Parameters	<ul style="list-style-type: none"> COMPx: Comparator instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: (1) Available only on COMP instance: COMP2.

Reference Manual to
LL API cross
reference:

- LL_COMP_POWERMODE_MEDIUMSPEED (1)
- LL_COMP_POWERMODE_ULTRALOWPOWER (1)

- COMP2_CSR SPEED LL_COMP_GetPowerMode

LL_COMP_SetInputPlus

Function name

**STATIC_INLINE void LL_COMP_SetInputPlus
(COMP_TypeDef * COMPx, uint32_t InputPlus)**

Function description

Set comparator input plus (non-inverting).

Parameters

- **COMPx:** Comparator instance
- **InputPlus:** This parameter can be one of the following values: (1) Available only on COMP instance: COMP1.
 - LL_COMP_INPUT_PLUS_NONE
 - LL_COMP_INPUT_PLUS_IO1 (2)
 - LL_COMP_INPUT_PLUS_IO2 (2)
 - LL_COMP_INPUT_PLUS_IO3 (2)(5)
 - LL_COMP_INPUT_PLUS_IO4 (2)(5)
 - LL_COMP_INPUT_PLUS_IO5 (1)
 - LL_COMP_INPUT_PLUS_IO6 (1)
 - LL_COMP_INPUT_PLUS_IO7 (1)
 - LL_COMP_INPUT_PLUS_IO8 (1)
 - LL_COMP_INPUT_PLUS_IO9 (1)
 - LL_COMP_INPUT_PLUS_IO10 (1)
 - LL_COMP_INPUT_PLUS_IO11 (1)
 - LL_COMP_INPUT_PLUS_IO12 (1)
 - LL_COMP_INPUT_PLUS_IO13 (1)
 - LL_COMP_INPUT_PLUS_IO14 (1)
 - LL_COMP_INPUT_PLUS_IO15 (1)
 - LL_COMP_INPUT_PLUS_IO16 (1)
 - LL_COMP_INPUT_PLUS_IO17 (1)
 - LL_COMP_INPUT_PLUS_IO18 (1)
 - LL_COMP_INPUT_PLUS_IO19 (1)
 - LL_COMP_INPUT_PLUS_IO20 (1)
 - LL_COMP_INPUT_PLUS_IO21 (1)
 - LL_COMP_INPUT_PLUS_IO22 (1)
 - LL_COMP_INPUT_PLUS_IO23 (1)
 - LL_COMP_INPUT_PLUS_IO24 (1)
 - LL_COMP_INPUT_PLUS_IO25 (1)
 - LL_COMP_INPUT_PLUS_IO26 (1)
 - LL_COMP_INPUT_PLUS_IO27 (1)
 - LL_COMP_INPUT_PLUS_IO28 (1)
 - LL_COMP_INPUT_PLUS_IO29 (1)(4)
 - LL_COMP_INPUT_PLUS_IO30 (1)(4)
 - LL_COMP_INPUT_PLUS_IO31 (1)(4)
 - LL_COMP_INPUT_PLUS_IO32 (1)(4)
 - LL_COMP_INPUT_PLUS_IO33 (1)(4)
 - LL_COMP_INPUT_PLUS_OPAMP1 (1)(3)
 - LL_COMP_INPUT_PLUS_OPAMP2 (1)(3)
 - LL_COMP_INPUT_PLUS_OPAMP3 (1)(4)

- (2) Available only on COMP instance: COMP2.
 - (3) Available on devices: STM32L100xB, STM32L151xB, STM32L152xB, STM32L100xBA, STM32L151xBA, STM32L152xBA, STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD
 - (4) Available on devices: STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD
 - (5) Available on devices: STM32L100xC, STM32L151xC, STM32L152xC, STM32L162xC, STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD || defined(STM32L151xE) || defined(STM32L151xDX, STM32L152xE, STM32L152xDX, STM32L162xE, STM32L162xDX)
- | | |
|---|--|
| Return values | • None: |
| Notes | <ul style="list-style-type: none"> • In case of comparator input selected to be connected to IO: GPIO pins are specific to each comparator instance. Refer to description of parameters or to reference manual. |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> • RI RI_ASCR1_CH LL_COMP_SetInputPlus • RI RI_ASCR2_GR6 LL_COMP_SetInputPlus |

LL_COMP_GetInputPlus

Function name	<code>_STATIC_INLINE uint32_t LL_COMP_GetInputPlus (COMP_TypeDef * COMPx)</code>
Function description	Get comparator input plus (non-inverting).
Parameters	<ul style="list-style-type: none"> • COMPx: Comparator instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (1) Available only on COMP instance: COMP1. <ul style="list-style-type: none"> - LL_COMP_INPUT_PLUS_NONE - LL_COMP_INPUT_PLUS_IO1 (2) - LL_COMP_INPUT_PLUS_IO2 (2) - LL_COMP_INPUT_PLUS_IO3 (2)(5) - LL_COMP_INPUT_PLUS_IO4 (2)(5) - LL_COMP_INPUT_PLUS_IO5 (1) - LL_COMP_INPUT_PLUS_IO6 (1) - LL_COMP_INPUT_PLUS_IO7 (1) - LL_COMP_INPUT_PLUS_IO8 (1) - LL_COMP_INPUT_PLUS_IO9 (1) - LL_COMP_INPUT_PLUS_IO10 (1) - LL_COMP_INPUT_PLUS_IO11 (1) - LL_COMP_INPUT_PLUS_IO12 (1) - LL_COMP_INPUT_PLUS_IO13 (1) - LL_COMP_INPUT_PLUS_IO14 (1) - LL_COMP_INPUT_PLUS_IO15 (1) - LL_COMP_INPUT_PLUS_IO16 (1) - LL_COMP_INPUT_PLUS_IO17 (1) - LL_COMP_INPUT_PLUS_IO18 (1) - LL_COMP_INPUT_PLUS_IO19 (1)

- LL_COMP_INPUT_PLUS_IO20 (1)
- LL_COMP_INPUT_PLUS_IO21 (1)
- LL_COMP_INPUT_PLUS_IO22 (1)
- LL_COMP_INPUT_PLUS_IO23 (1)
- LL_COMP_INPUT_PLUS_IO24 (1)
- LL_COMP_INPUT_PLUS_IO25 (1)
- LL_COMP_INPUT_PLUS_IO26 (1)
- LL_COMP_INPUT_PLUS_IO27 (1)
- LL_COMP_INPUT_PLUS_IO28 (1)
- LL_COMP_INPUT_PLUS_IO29 (1)(4)
- LL_COMP_INPUT_PLUS_IO30 (1)(4)
- LL_COMP_INPUT_PLUS_IO31 (1)(4)
- LL_COMP_INPUT_PLUS_IO32 (1)(4)
- LL_COMP_INPUT_PLUS_IO33 (1)(4)
- LL_COMP_INPUT_PLUS_OPAMP1 (1)(3)
- LL_COMP_INPUT_PLUS_OPAMP2 (1)(3)
- LL_COMP_INPUT_PLUS_OPAMP3 (1)(4)
- (2) Available only on COMP instance: COMP2.
- (3) Available on devices: STM32L100xB, STM32L151xB, STM32L152xB, STM32L100xBA, STM32L151xBA, STM32L152xBA, STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD
- (4) Available on devices: STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD
- (5) Available on devices: STM32L100xC, STM32L151xC, STM32L152xC, STM32L162xC, STM32L151xCA, STM32L151xD, STM32L152xCA, STM32L152xD, STM32L162xCA, STM32L162xD || defined(STM32L151xE) || defined(STM32L151xDX, STM32L152xE, STM32L152xDX, STM32L162xE, STM32L162xDX)

Notes

- In case of comparator input selected to be connected to IO: GPIO pins are specific to each comparator instance. Refer to description of parameters or to reference manual.

**Reference Manual to
LL API cross
reference:**

- RI RI_ASCR1_CH LL_COMP_GetInputPlus
- RI RI_ASCR2_GR6 LL_COMP_GetInputPlus

LL_COMP_SetInputMinus**Function name**

```
__STATIC_INLINE void LL_COMP_SetInputMinus
(COMP_TypeDef * COMPx, uint32_t InputMinus)
```

Function description

Set comparator input minus (inverting).

Parameters

- **COMPx:** Comparator instance
- **InputMinus:** This parameter can be one of the following values: (1) Available only on COMP instance: COMP2.
 - LL_COMP_INPUT_MINUS_1_4VREFINT (1)
 - LL_COMP_INPUT_MINUS_1_2VREFINT (1)
 - LL_COMP_INPUT_MINUS_3_4VREFINT (1)
 - LL_COMP_INPUT_MINUS_VREFINT
 - LL_COMP_INPUT_MINUS_DAC1_CH1 (1)



	<ul style="list-style-type: none"> - LL_COMP_INPUT_MINUS_DAC1_CH2 (1) - LL_COMP_INPUT_MINUS_IO1 (1)
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • In case of comparator input selected to be connected to IO: GPIO pins are specific to each comparator instance. Refer to description of parameters or to reference manual.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR COMP_CSR_INSEL LL_COMP_SetInputMinus

LL_COMP_GetInputMinus

Function name	<code>__STATIC_INLINE uint32_t LL_COMP_GetInputMinus(COMP_TypeDef * COMPx)</code>
Function description	Get comparator input minus (inverting).
Parameters	<ul style="list-style-type: none"> • COMPx: Comparator instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (1) Available only on COMP instance: COMP2. <ul style="list-style-type: none"> - LL_COMP_INPUT_MINUS_1_4VREFINT (1) - LL_COMP_INPUT_MINUS_1_2VREFINT (1) - LL_COMP_INPUT_MINUS_3_4VREFINT (1) - LL_COMP_INPUT_MINUS_VREFINT - LL_COMP_INPUT_MINUS_DAC1_CH1 (1) - LL_COMP_INPUT_MINUS_DAC1_CH2 (1) - LL_COMP_INPUT_MINUS_IO1 (1)
Notes	<ul style="list-style-type: none"> • In case of comparator input selected to be connected to IO: GPIO pins are specific to each comparator instance. Refer to description of parameters or to reference manual.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR COMP_CSR_INSEL LL_COMP_SetInputMinus

LL_COMP_SetInputPullingResistor

Function name	<code>__STATIC_INLINE void LL_COMP_SetInputPullingResistor(COMP_TypeDef * COMPx, uint32_t InputPullingResistor)</code>
Function description	Set comparator input pulling resistor.
Parameters	<ul style="list-style-type: none"> • COMPx: Comparator instance • InputPullingResistor: This parameter can be one of the following values: (1) Available only on COMP instance: COMP1. <ul style="list-style-type: none"> - LL_COMP_INPUT_MINUS_PULL_NO - LL_COMP_INPUT_MINUS_PULL_UP_10K (1) - LL_COMP_INPUT_MINUS_PULL_UP_400K (1) - LL_COMP_INPUT_MINUS_PULL_DOWN_10K (1) - LL_COMP_INPUT_MINUS_PULL_DOWN_400K (1)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to	<ul style="list-style-type: none"> • CSR 10KPU LL_COMP_SetInputPullingResistor

- LL API cross reference:
- CSR 400KPU LL_COMP_SetInputPullingResistor
 - CSR 10KPD LL_COMP_SetInputPullingResistor
 - CSR 400KPD LL_COMP_SetInputPullingResistor

LL_COMP_GetInputPullingResistor

Function name	<code>__STATIC_INLINE uint32_t LL_COMP_GetInputPullingResistor (COMP_TypeDef * COMPx)</code>
Function description	Get comparator input pulling resistor.
Parameters	<ul style="list-style-type: none"> • COMPx: Comparator instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (1) Available only on COMP instance: COMP1. <ul style="list-style-type: none"> - LL_COMP_INPUT_MINUS_PULL_NO - LL_COMP_INPUT_MINUS_PULL_UP_10K (1) - LL_COMP_INPUT_MINUS_PULL_UP_400K (1) - LL_COMP_INPUT_MINUS_PULL_DOWN_10K (1) - LL_COMP_INPUT_MINUS_PULL_DOWN_400K (1)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR 10KPU LL_COMP_SetInputPullingResistor • CSR 400KPU LL_COMP_SetInputPullingResistor • CSR 10KPD LL_COMP_SetInputPullingResistor • CSR 400KPD LL_COMP_SetInputPullingResistor

LL_COMP_SetOutputSelection

Function name	<code>__STATIC_INLINE void LL_COMP_SetOutputSelection (COMP_TypeDef * COMPx, uint32_t OutputSelection)</code>
Function description	Set comparator output selection.
Parameters	<ul style="list-style-type: none"> • COMPx: Comparator instance • OutputSelection: This parameter can be one of the following values: (1) Parameter availability depending on timer availability on the selected device. (2) Available only on COMP instance: COMP2. <ul style="list-style-type: none"> - LL_COMP_OUTPUT_NONE - LL_COMP_OUTPUT_TIM2_IC4 (1)(2) - LL_COMP_OUTPUT_TIM2_OCREFCLR (1)(2) - LL_COMP_OUTPUT_TIM3_IC4 (1)(2) - LL_COMP_OUTPUT_TIM3_OCREFCLR (1)(2) - LL_COMP_OUTPUT_TIM4_IC4 (1)(2) - LL_COMP_OUTPUT_TIM4_OCREFCLR (1)(2) - LL_COMP_OUTPUT_TIM10_IC1 (1)(2)
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Availability of parameters of output selection to timer depends on timers availability on the selected device.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR OUTSEL LL_COMP_SetOutputSelection

LL_COMP_GetOutputSelection

Function name	<code>_STATIC_INLINE uint32_t LL_COMP_GetOutputSelection (COMP_TypeDef * COMPx)</code>
Function description	Get comparator output selection.
Parameters	<ul style="list-style-type: none"> • COMPx: Comparator instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (1) Parameter availability depending on timer availability on the selected device. (2) Available only on COMP instance: COMP2. <ul style="list-style-type: none"> - <code>LL_COMP_OUTPUT_NONE</code> - <code>LL_COMP_OUTPUT_TIM2_IC4</code> (1)(2) - <code>LL_COMP_OUTPUT_TIM2_OCREFCLR</code> (1)(2) - <code>LL_COMP_OUTPUT_TIM3_IC4</code> (1)(2) - <code>LL_COMP_OUTPUT_TIM3_OCREFCLR</code> (1)(2) - <code>LL_COMP_OUTPUT_TIM4_IC4</code> (1)(2) - <code>LL_COMP_OUTPUT_TIM4_OCREFCLR</code> (1)(2) - <code>LL_COMP_OUTPUT_TIM10_IC1</code> (1)(2)
Notes	<ul style="list-style-type: none"> • Availability of parameters of output selection to timer depends on timers availability on the selected device.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR OUTSEL LL_COMP_GetOutputSelection

LL_COMP_Enable

Function name	<code>_STATIC_INLINE void LL_COMP_Enable (COMP_TypeDef * COMPx)</code>
Function description	Enable comparator instance.
Parameters	<ul style="list-style-type: none"> • COMPx: Comparator instance (1)
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • After enable from off state, comparator requires a delay to reach propagation delay specification. Refer to device datasheet, parameter "tSTART".
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR COMP1EN LL_COMP_Enable • CSR COMP_CSR_INSEL LL_COMP_Enable

LL_COMP_Disable

Function name	<code>_STATIC_INLINE void LL_COMP_Disable (COMP_TypeDef * COMPx)</code>
Function description	Disable comparator instance.
Parameters	<ul style="list-style-type: none"> • COMPx: Comparator instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • On this STM32 serie, COMP2 is disabled by clearing input minus selection. If COMP2 must be enabled afterwards, input

Reference Manual to LL API cross reference:	minus must be set. Refer to function LL_COMP_SetInputMinus() .
	<ul style="list-style-type: none"> • CSR COMP1EN LL_COMP_Disable • CSR COMP_CSR_INSEL LL_COMP_Disable

LL_COMP_IsEnabled

Function name Function description Parameters Return values Reference Manual to LL API cross reference:	_STATIC_INLINE uint32_t LL_COMP_IsEnabled (COMP_TypeDef * COMPx) <p>Get comparator enable state (0: COMP is disabled, 1: COMP is enabled)</p> <ul style="list-style-type: none"> • COMPx: Comparator instance • State: of bit (1 or 0). <ul style="list-style-type: none"> • CSR COMP1EN LL_COMP_IsEnabled • CSR COMP_CSR_INSEL LL_COMP_IsEnabled
---	--

LL_COMP_ReadOutputLevel

Function name Function description Parameters Return values Notes Reference Manual to LL API cross reference:	_STATIC_INLINE uint32_t LL_COMP_ReadOutputLevel (COMP_TypeDef * COMPx) <p>Read comparator instance output level.</p> <ul style="list-style-type: none"> • COMPx: Comparator instance • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_COMP_OUTPUT_LEVEL_LOW – LL_COMP_OUTPUT_LEVEL_HIGH <ul style="list-style-type: none"> • On this STM32 serie, comparator polarity is not settable and not inverted: Comparator output is low when the input plus is at a lower voltage than the input minusComparator output is high when the input plus is at a higher voltage than the input minus <ul style="list-style-type: none"> • CSR CMP1OUT LL_COMP_ReadOutputLevel • CSR CMP2OUT LL_COMP_ReadOutputLevel
--	---

LL_COMP_DeInit

Function name Function description Parameters Return values Notes	ErrorStatus LL_COMP_DeInit (COMP_TypeDef * COMPx) <p>De-initialize registers of the selected COMP instance to their default reset values.</p> <ul style="list-style-type: none"> • COMPx: COMP instance • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: COMP registers are de-initialized – ERROR: COMP registers are not de-initialized • If comparator is locked, de-initialization by software is not possible. The only way to unlock the comparator is a device
---	--

hardware reset.

LL_COMP_Init

Function name	ErrorStatus LL_COMP_Init (COMP_TypeDef * COMPx, LL_COMP_InitTypeDef * COMP_InitStruct)
Function description	Initialize some features of COMP instance.
Parameters	<ul style="list-style-type: none"> • COMPx: COMP instance • COMP_InitStruct: Pointer to a LL_COMP_InitTypeDef structure
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: COMP registers are initialized – ERROR: COMP registers are not initialized
Notes	<ul style="list-style-type: none"> • This function configures features of the selected COMP instance. Some features are also available at scope COMP common instance (common to several COMP instances). Refer to functions having argument "COMPxy_COMMON" as parameter.

LL_COMP_StructInit

Function name	void LL_COMP_StructInit (LL_COMP_InitTypeDef * COMP_InitStruct)
Function description	Set each LL_COMP_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • COMP_InitStruct: pointer to a LL_COMP_InitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> • None:

50.3 COMP Firmware driver defines

50.3.1 COMP

Comparator common modes - Window mode

LL_COMP_WINDOWMODE_DISABLE	Window mode disable: Comparators 1 and 2 are independent
LL_COMP_WINDOWMODE_COMP2_INPUT_PLUS_COMMON	Window mode enable: Comparators instances pair COMP1 and COMP2 have their input plus connected together. The common input is COMP2 input plus (COMP1 input plus is no more accessible, either from GPIO and from ADC channel VCOMP).

Definitions of COMP hardware constraints delays

`LL_COMP_DELAY_STARTUP_US` Delay for COMP startup time

Comparator inputs - Input minus (input inverting) selection

<code>LL_COMP_INPUT_MINUS_1_4VREFINT</code>	Comparator input minus connected to 1/4 VrefInt (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_MINUS_1_2VREFINT</code>	Comparator input minus connected to 1/2 VrefInt (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_MINUS_3_4VREFINT</code>	Comparator input minus connected to 3/4 VrefInt (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_MINUS_VREFINT</code>	Comparator input minus connected to VrefInt
<code>LL_COMP_INPUT_MINUS_DAC1_CH1</code>	Comparator input minus connected to DAC1 channel 1 (DAC_OUT1) (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_MINUS_DAC1_CH2</code>	Comparator input minus connected to DAC1 channel 2 (DAC_OUT2) (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_MINUS_IO1</code>	Comparator input minus connected to IO1 (pin PB3 for COMP2) (specific to COMP instance: COMP2)

Comparator inputs - Input plus (input non-inverting) selection

<code>LL_COMP_INPUT_PLUS_NONE</code>	Comparator input plus connected not connected
<code>LL_COMP_INPUT_PLUS_IO1</code>	Comparator input plus connected to IO1 (pin PB4 for COMP2) (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_PLUS_IO2</code>	Comparator input plus connected to IO1 (pin PB5 for COMP2) (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_PLUS_IO3</code>	Comparator input plus connected to IO1 (pin PB6 for COMP2) (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_PLUS_IO4</code>	Comparator input plus connected to IO1 (pin PB7 for COMP2) (specific to COMP instance: COMP2)
<code>LL_COMP_INPUT_PLUS_IO5</code>	Comparator input plus connected to IO5 (pin PA0 for COMP1) (specific to COMP instance: COMP1)
<code>LL_COMP_INPUT_PLUS_IO6</code>	Comparator input plus connected to IO6 (pin PA1 for COMP1) (specific to COMP instance: COMP1)
<code>LL_COMP_INPUT_PLUS_IO7</code>	Comparator input plus connected to IO7 (pin PA2 for COMP1) (specific to COMP instance: COMP1)
<code>LL_COMP_INPUT_PLUS_IO8</code>	Comparator input plus connected to IO8 (pin PA3 for COMP1) (specific to COMP instance: COMP1)
<code>LL_COMP_INPUT_PLUS_IO9</code>	Comparator input plus connected to IO9 (pin PA4 for COMP1) (specific to COMP instance: COMP1)
<code>LL_COMP_INPUT_PLUS_IO10</code>	Comparator input plus connected to IO10 (pin PA5 for COMP1) (specific to COMP instance: COMP1)
<code>LL_COMP_INPUT_PLUS_IO11</code>	Comparator input plus connected to IO11 (pin PA5 for COMP1) (specific to COMP instance: COMP1)
<code>LL_COMP_INPUT_PLUS_IO12</code>	Comparator input plus connected to IO12 (pin PA7 for COMP1) (specific to COMP instance: COMP1)

LL_COMP_INPUT_PLUS_IO13	Comparator input plus connected to IO13 (pin PB0 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO14	Comparator input plus connected to IO14 (pin PB1 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO15	Comparator input plus connected to IO15 (pin PC0 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO16	Comparator input plus connected to IO16 (pin PC1 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO17	Comparator input plus connected to IO17 (pin PC2 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO18	Comparator input plus connected to IO18 (pin PC3 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO19	Comparator input plus connected to IO19 (pin PC4 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO20	Comparator input plus connected to IO20 (pin PC5 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO21	Comparator input plus connected to IO21 (pin PB12 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO22	Comparator input plus connected to IO22 (pin PB13 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO23	Comparator input plus connected to IO23 (pin PB14 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO24	Comparator input plus connected to IO24 (pin PB15 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO25	Comparator input plus connected to IO25 (pin PE7 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO26	Comparator input plus connected to IO26 (pin PE8 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO27	Comparator input plus connected to IO27 (pin PE9 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO28	Comparator input plus connected to IO28 (pin PE10 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO29	Comparator input plus connected to IO29 (pin PF6 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO30	Comparator input plus connected to IO30 (pin PF7 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO31	Comparator input plus connected to IO31 (pin PF8 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO32	Comparator input plus connected to IO32 (pin PF9 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_IO33	Comparator input plus connected to IO33 (pin PF10 for COMP1) (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_OPAMP1	Comparator input plus connected to OPAMP1 output (specific to COMP instance: COMP1)

LL_COMP_INPUT_PLUS_OPAMP2	Comparator input plus connected to OPAMP2 output (specific to COMP instance: COMP1)
LL_COMP_INPUT_PLUS_OPAMP3	Comparator input plus connected to OPAMP3 output (specific to COMP instance: COMP1)
Comparator input - Pulling resistor	
LL_COMP_INPUT_MINUS_PULL_NO	Comparator input minus not connected to any pulling resistor
LL_COMP_INPUT_MINUS_PULL_UP_10K	Comparator input minus connected to pull-up resistor of 10kOhm (specific to COMP instance: COMP1)
LL_COMP_INPUT_MINUS_PULL_UP_400K	Comparator input minus connected to pull-up resistor of 400kOhm (specific to COMP instance: COMP1)
LL_COMP_INPUT_MINUS_PULL_DOWN_10K	Comparator input minus connected to pull-down resistor of 10kOhm (specific to COMP instance: COMP1)
LL_COMP_INPUT_MINUS_PULL_DOWN_400K	Comparator input minus connected to pull-down resistor of 400kOhm (specific to COMP instance: COMP1)
Comparator output - Output level	
LL_COMP_OUTPUT_LEVEL_LOW	Comparator output level low (if the polarity is not inverted, otherwise to be complemented)
LL_COMP_OUTPUT_LEVEL_HIGH	Comparator output level high (if the polarity is not inverted, otherwise to be complemented)
Comparator output - Output selection	
LL_COMP_OUTPUT_NONE	COMP output is not connected to other peripherals (except GPIO and EXTI that are always connected to COMP output) (specific to COMP instance: COMP2)
LL_COMP_OUTPUT_TIM2_IC4	COMP output connected to TIM2 input capture 4 (specific to COMP instance: COMP2)
LL_COMP_OUTPUT_TIM2_OCREFCLR	COMP output connected to TIM2 OCREF clear (specific to COMP instance: COMP2)
LL_COMP_OUTPUT_TIM3_IC4	COMP output connected to TIM3 input capture 4 (specific to COMP instance: COMP2)
LL_COMP_OUTPUT_TIM3_OCREFCLR	COMP output connected to TIM3 OCREF clear (specific to COMP instance: COMP2)
LL_COMP_OUTPUT_TIM4_IC4	COMP output connected to TIM4 input capture 4 (specific to COMP instance: COMP2)
LL_COMP_OUTPUT_TIM4_OCREFCLR	COMP output connected to TIM4 OCREF clear (specific to COMP instance: COMP2)
LL_COMP_OUTPUT_TIM10_IC1	COMP output connected to TIM10 input capture 1 (specific to COMP instance: COMP2)
Comparator modes - Power mode	

LL_COMP_POWERMODE_ULTRALOWPOWER	COMP power mode to low speed (specific to COMP instance: COMP2)
LL_COMP_POWERMODE_MEDIUMSPEED	COMP power mode to fast speed (specific to COMP instance: COMP2)

COMP helper macro

<u>LL_COMP_COMMON_INSTANCE</u>	Description: <ul style="list-style-type: none">Helper macro to select the COMP common instance to which is belonging the selected COMP instance. Parameters: <ul style="list-style-type: none"><u>COMPx</u>: COMP instance Return value: <ul style="list-style-type: none">COMP: common instance or value "0" if there is no COMP common instance. Notes: <ul style="list-style-type: none">COMP common register instance can be used to set parameters common to several COMP instances. Refer to functions having argument "COMPxy_COMMON" as parameter.
--------------------------------	--

Common write and read registers macro

<u>LL_COMP_WriteReg</u>	Description: <ul style="list-style-type: none">Write a value in COMP register. Parameters: <ul style="list-style-type: none"><u>INSTANCE</u>: comparator instance<u>REG</u>: Register to be written<u>VALUE</u>: Value to be written in the register Return value: <ul style="list-style-type: none">None
<u>LL_COMP_ReadReg</u>	Description: <ul style="list-style-type: none">Read a value in COMP register. Parameters: <ul style="list-style-type: none"><u>INSTANCE</u>: comparator instance<u>REG</u>: Register to be read Return value: <ul style="list-style-type: none">Register: value

51 LL CORTEX Generic Driver

51.1 CORTEX Firmware driver API description

51.1.1 Detailed description of functions

LL_SYSTICK_IsActiveCounterFlag

Function name **`__STATIC_INLINE uint32_t LL_SYSTICK_IsActiveCounterFlag(void)`**

Function description This function checks if the Systick counter flag is active or not.

Return values • **State:** of bit (1 or 0).

Notes • It can be used in timeout function on application side.

Reference Manual to
LL API cross
reference: • STK_CTRL COUNTFLAG LL_SYSTICK_IsActiveCounterFlag

LL_SYSTICK_SetClkSource

Function name **`__STATIC_INLINE void LL_SYSTICK_SetClkSource(uint32_t Source)`**

Function description Configures the SysTick clock source.

Parameters • **Source:** This parameter can be one of the following values:
– LL_SYSTICK_CLKSOURCE_HCLK_DIV8
– LL_SYSTICK_CLKSOURCE_HCLK

Return values • **None:**

Reference Manual to
LL API cross
reference: • STK_CTRL CLKSOURCE LL_SYSTICK_SetClkSource

LL_SYSTICK_GetClkSource

Function name **`__STATIC_INLINE uint32_t LL_SYSTICK_GetClkSource(void)`**

Function description Get the SysTick clock source.

Return values • **Returned:** value can be one of the following values:
– LL_SYSTICK_CLKSOURCE_HCLK_DIV8
– LL_SYSTICK_CLKSOURCE_HCLK

Reference Manual to
LL API cross
reference: • STK_CTRL CLKSOURCE LL_SYSTICK_GetClkSource

LL_SYSTICK_EnableIT

Function name **`__STATIC_INLINE void LL_SYSTICK_EnableIT(void)`**

Function description Enable SysTick exception request.

Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • STK_CTRL TICKINT LL_SYSTICK_EnableIT

LL_SYSTICK_DisableIT

Function name	<code>__STATIC_INLINE void LL_SYSTICK_DisableIT (void)</code>
Function description	Disable SysTick exception request.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • STK_CTRL TICKINT LL_SYSTICK_DisableIT

LL_SYSTICK_IsEnabledIT

Function name	<code>__STATIC_INLINE uint32_t LL_SYSTICK_IsEnabledIT (void)</code>
Function description	Checks if the SYSTICK interrupt is enabled or disabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • STK_CTRL TICKINT LL_SYSTICK_IsEnabledIT

LL_LPM_EnableSleep

Function name	<code>__STATIC_INLINE void LL_LPM_EnableSleep (void)</code>
Function description	Processor uses sleep as its low power mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SCB_SCR SLEEPDEEP LL_LPM_EnableSleep

LL_LPM_EnableDeepSleep

Function name	<code>__STATIC_INLINE void LL_LPM_EnableDeepSleep (void)</code>
Function description	Processor uses deep sleep as its low power mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SCB_SCR SLEEPDEEP LL_LPM_EnableDeepSleep

LL_LPM_EnableSleepOnExit

Function name	<code>__STATIC_INLINE void LL_LPM_EnableSleepOnExit (void)</code>
Function description	Configures sleep-on-exit when returning from Handler mode to Thread mode.

Return values	<ul style="list-style-type: none">None:
Notes	<ul style="list-style-type: none">Setting this bit to 1 enables an interrupt-driven application to avoid returning to an empty main application.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">SCB_SCR SLEEPONEXIT LL_LPM_EnableSleepOnExit

LL_LPM_DisableSleepOnExit

Function name	<code>__STATIC_INLINE void LL_LPM_DisableSleepOnExit (void)</code>
Function description	Do not sleep when returning to Thread mode.
Return values	<ul style="list-style-type: none">None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">SCB_SCR SLEEPONEXIT LL_LPM_DisableSleepOnExit

LL_LPM_EnableEventOnPend

Function name	<code>__STATIC_INLINE void LL_LPM_EnableEventOnPend (void)</code>
Function description	Enabled events and all interrupts, including disabled interrupts, can wakeup the processor.
Return values	<ul style="list-style-type: none">None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">SCB_SCR SEVEONPEND LL_LPM_EnableEventOnPend

LL_LPM_DisableEventOnPend

Function name	<code>__STATIC_INLINE void LL_LPM_DisableEventOnPend (void)</code>
Function description	Only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded.
Return values	<ul style="list-style-type: none">None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">SCB_SCR SEVEONPEND LL_LPM_DisableEventOnPend

LL_HANDLER_EnableFault

Function name	<code>__STATIC_INLINE void LL_HANDLER_EnableFault (uint32_t Fault)</code>
Function description	Enable a fault in System handler control register (SHCSR)
Parameters	<ul style="list-style-type: none">Fault: This parameter can be a combination of the following values:<ul style="list-style-type: none">LL_HANDLER_FAULT_USGLL_HANDLER_FAULT_BUSLL_HANDLER_FAULT_MEM

Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SCB_SHCSR MEMFAULTENA LL_HANDLER_EnableFault

LL_HANDLER_DisableFault

Function name	<code>__STATIC_INLINE void LL_HANDLER_DisableFault (uint32_t Fault)</code>
Function description	Disable a fault in System handler control register (SHCSR)
Parameters	<ul style="list-style-type: none"> Fault: This parameter can be a combination of the following values: <ul style="list-style-type: none"> LL_HANDLER_FAULT_USG LL_HANDLER_FAULT_BUS LL_HANDLER_FAULT_MEM
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SCB_SHCSR MEMFAULTENA LL_HANDLER_DisableFault

LL_CPUID_GetImplementer

Function name	<code>__STATIC_INLINE uint32_t LL_CPUID_GetImplementer (void)</code>
Function description	Get Implementer code.
Return values	<ul style="list-style-type: none"> Value: should be equal to 0x41 for ARM
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SCB_CPUID IMPLEMENTER LL_CPUID_GetImplementer

LL_CPUID_GetVariant

Function name	<code>__STATIC_INLINE uint32_t LL_CPUID_GetVariant (void)</code>
Function description	Get Variant number (The r value in the rnpr product revision identifier)
Return values	<ul style="list-style-type: none"> Value: between 0 and 255 (0x1: revision 1, 0x2: revision 2)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SCB_CPUID VARIANT LL_CPUID_GetVariant

LL_CPUID_GetConstant

Function name	<code>__STATIC_INLINE uint32_t LL_CPUID_GetConstant (void)</code>
Function description	Get Constant number.
Return values	<ul style="list-style-type: none"> Value: should be equal to 0xF for Cortex-M3 devices
Reference Manual to LL API cross	<ul style="list-style-type: none"> SCB_CPUID ARCHITECTURE LL_CPUID_GetConstant

reference:

LL_CPUID_GetParNo

Function name **__STATIC_INLINE uint32_t LL_CPUID_GetParNo (void)**

Function description Get Part number.

Return values • **Value:** should be equal to 0xC23 for Cortex-M3

Reference Manual to
LL API cross
reference:
SCB_CPUID PARTNO LL_CPUID_GetParNo

LL_CPUID_GetRevision

Function name **__STATIC_INLINE uint32_t LL_CPUID_GetRevision (void)**

Function description Get Revision number (The p value in the rnpn product revision identifier, indicates patch release)

Return values • **Value:** between 0 and 255 (0x0: patch 0, 0x1: patch 1)

Reference Manual to
LL API cross
reference:
SCB_CPUID REVISION LL_CPUID_GetRevision

LL_MPU_Enable

Function name **__STATIC_INLINE void LL_MPU_Enable (uint32_t Options)**

Function description Enable MPU with input options.

Parameters • **Options:** This parameter can be one of the following values:
 - LL_MPU_CTRL_HFNMI_PRIVDEF_NONE
 - LL_MPU_CTRL_HARDFAULT_NMI
 - LL_MPU_CTRL_PRIVILEGED_DEFAULT
 - LL_MPU_CTRL_HFNMI_PRIVDEF

Return values • **None:**

Reference Manual to
LL API cross
reference:
MPU_CTRL ENABLE LL_MPU_Enable

LL_MPU_Disable

Function name **__STATIC_INLINE void LL_MPU_Disable (void)**

Function description Disable MPU.

Return values • **None:**

Reference Manual to
LL API cross
reference:
MPU_CTRL ENABLE LL_MPU_Disable

LL_MPU_IsEnabled

Function name **__STATIC_INLINE uint32_t LL_MPU_IsEnabled (void)**

Function description	Check if MPU is enabled or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	MPU_CTRL ENABLE LL_MPU_IsEnabled

LL_MPU_EnableRegion

Function name	<code>__STATIC_INLINE void LL_MPU_EnableRegion (uint32_t Region)</code>
Function description	Enable a MPU region.
Parameters	<ul style="list-style-type: none"> • Region: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_MPU_REGION_NUMBER0 – LL_MPU_REGION_NUMBER1 – LL_MPU_REGION_NUMBER2 – LL_MPU_REGION_NUMBER3 – LL_MPU_REGION_NUMBER4 – LL_MPU_REGION_NUMBER5 – LL_MPU_REGION_NUMBER6 – LL_MPU_REGION_NUMBER7
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	MPU_RASR ENABLE LL_MPU_EnableRegion

LL_MPU_ConfigRegion

Function name	<code>__STATIC_INLINE void LL_MPU_ConfigRegion (uint32_t Region, uint32_t SubRegionDisable, uint32_t Address, uint32_t Attributes)</code>
Function description	Configure and enable a region.
Parameters	<ul style="list-style-type: none"> • Region: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_MPU_REGION_NUMBER0 – LL_MPU_REGION_NUMBER1 – LL_MPU_REGION_NUMBER2 – LL_MPU_REGION_NUMBER3 – LL_MPU_REGION_NUMBER4 – LL_MPU_REGION_NUMBER5 – LL_MPU_REGION_NUMBER6 – LL_MPU_REGION_NUMBER7 • Address: Value of region base address • SubRegionDisable: Sub-region disable value between Min_Data = 0x00 and Max_Data = 0xFF • Attributes: This parameter can be a combination of the following values: <ul style="list-style-type: none"> – LL_MPU_REGION_SIZE_32B or LL_MPU_REGION_SIZE_64B or LL_MPU_REGION_SIZE_128B or LL_MPU_REGION_SIZE_256B or LL_MPU_REGION_SIZE_512B or

- LL_MPU_REGION_SIZE_1KB or
 LL_MPU_REGION_SIZE_2KB or
 LL_MPU_REGION_SIZE_4KB or
 LL_MPU_REGION_SIZE_8KB or
 LL_MPU_REGION_SIZE_16KB or
 LL_MPU_REGION_SIZE_32KB or
 LL_MPU_REGION_SIZE_64KB or
 LL_MPU_REGION_SIZE_128KB or
 LL_MPU_REGION_SIZE_256KB or
 LL_MPU_REGION_SIZE_512KB or
 LL_MPU_REGION_SIZE_1MB or
 LL_MPU_REGION_SIZE_2MB or
 LL_MPU_REGION_SIZE_4MB or
 LL_MPU_REGION_SIZE_8MB or
 LL_MPU_REGION_SIZE_16MB or
 LL_MPU_REGION_SIZE_32MB or
 LL_MPU_REGION_SIZE_64MB or
 LL_MPU_REGION_SIZE_128MB or
 LL_MPU_REGION_SIZE_256MB or
 LL_MPU_REGION_SIZE_512MB or
 LL_MPU_REGION_SIZE_1GB or
 LL_MPU_REGION_SIZE_2GB or
 LL_MPU_REGION_SIZE_4GB
 - LL_MPU_REGION_NO_ACCESS or
 LL_MPU_REGION_PRIV_RW or
 LL_MPU_REGION_PRIV_RW_URO or
 LL_MPU_REGION_FULL_ACCESS or
 LL_MPU_REGION_PRIV_RO or
 LL_MPU_REGION_PRIV_RO_URO
 - LL_MPU_TEX_LEVEL0 or LL_MPU_TEX_LEVEL1 or
 LL_MPU_TEX_LEVEL2 or LL_MPU_TEX_LEVEL4
 - LL_MPU_INSTRUCTION_ACCESS_ENABLE or
 LL_MPU_INSTRUCTION_ACCESS_DISABLE
 - LL_MPU_ACCESS_SHAREABLE or
 LL_MPU_ACCESS_NOT_SHAREABLE
 - LL_MPU_ACCESS_CACHEABLE or
 LL_MPU_ACCESS_NOT_CACHEABLE
 - LL_MPU_ACCESS_BUFFERABLE or
 LL_MPU_ACCESS_NOT_BUFFERABLE

Return values

Reference Manual to
LL API cross
reference:

- **None:**
- MPU_RNR REGION LL_MPU_ConfigRegion
- MPU_RBAR REGION LL_MPU_ConfigRegion
- MPU_RBAR ADDR LL_MPU_ConfigRegion
- MPU_RASR XN LL_MPU_ConfigRegion
- MPU_RASR AP LL_MPU_ConfigRegion
- MPU_RASR S LL_MPU_ConfigRegion
- MPU_RASR C LL_MPU_ConfigRegion
- MPU_RASR B LL_MPU_ConfigRegion
- MPU_RASR SIZE LL_MPU_ConfigRegion

LL_MPU_DisableRegion

Function name **_STATIC_INLINE void LL_MPU_DisableRegion (uint32_t**

Region)	
Function description	Disable a region.
Parameters	<ul style="list-style-type: none"> • Region: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_MPURREGION_NUMBER0 – LL_MPURREGION_NUMBER1 – LL_MPURREGION_NUMBER2 – LL_MPURREGION_NUMBER3 – LL_MPURREGION_NUMBER4 – LL_MPURREGION_NUMBER5 – LL_MPURREGION_NUMBER6 – LL_MPURREGION_NUMBER7
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • MPU_RNR REGION LL_MPUDisableRegion • MPU_RASR ENABLE LL_MPUDisableRegion

51.2 CORTEX Firmware driver defines

51.2.1 CORTEX

MPU Bufferable Access

LL_MPUDACCESS_BUFFERABLE Bufferable memory attribute

LL_MPUDACCESS_NOT_BUFFERABLE Not Bufferable memory attribute

MPU Cacheable Access

LL_MPUDACCESS_CACHEABLE Cacheable memory attribute

LL_MPUDACCESS_NOT_CACHEABLE Not Cacheable memory attribute

SYSTICK Clock Source

LL_SYSTICK_CLKSOURCE_HCLK_DIV8 AHB clock divided by 8 selected as SysTick clock source.

LL_SYSTICK_CLKSOURCE_HCLK AHB clock selected as SysTick clock source.

MPU Control

LL_MPUDCTRL_HFNMI_PRIVDEF_NONE Disable NMI and privileged SW access

LL_MPUDCTRL_HARDFAULT_NMI Enables the operation of MPU during hard fault, NMI, and FAULTMASK handlers

LL_MPUDCTRL_PRIVILEGED_DEFAULT Enable privileged software access to default memory map

LL_MPUDCTRL_HFNMI_PRIVDEF Enable NMI and privileged SW access

Handler Fault type

LL_HANDLER_FAULT_USG Usage fault

LL_HANDLER_FAULT_BUS Bus fault

LL_HANDLER_FAULT_MEM Memory management fault

MPU Instruction Access

<code>LL_MPU_INSTRUCTION_ACCESS_ENABLE</code>	Instruction fetches enabled
<code>LL_MPU_INSTRUCTION_ACCESS_DISABLE</code>	Instruction fetches disabled

MPU Region Number

<code>LL_MPU_REGION_NUMBER0</code>	REGION Number 0
<code>LL_MPU_REGION_NUMBER1</code>	REGION Number 1
<code>LL_MPU_REGION_NUMBER2</code>	REGION Number 2
<code>LL_MPU_REGION_NUMBER3</code>	REGION Number 3
<code>LL_MPU_REGION_NUMBER4</code>	REGION Number 4
<code>LL_MPU_REGION_NUMBER5</code>	REGION Number 5
<code>LL_MPU_REGION_NUMBER6</code>	REGION Number 6
<code>LL_MPU_REGION_NUMBER7</code>	REGION Number 7

MPU Region Privileges

<code>LL_MPU_REGION_NO_ACCESS</code>	No access
<code>LL_MPU_REGION_PRIV_RW</code>	RW privileged (privileged access only)
<code>LL_MPU_REGION_PRIV_RW_URO</code>	RW privileged - RO user (Write in a user program generates a fault)
<code>LL_MPU_REGION_FULL_ACCESS</code>	RW privileged & user (Full access)
<code>LL_MPU_REGION_PRIV_RO</code>	RO privileged (privileged read only)
<code>LL_MPU_REGION_PRIV_RO_URO</code>	RO privileged & user (read only)

MPU Region Size

<code>LL_MPU_REGION_SIZE_32B</code>	32B Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_64B</code>	64B Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_128B</code>	128B Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_256B</code>	256B Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_512B</code>	512B Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_1KB</code>	1KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_2KB</code>	2KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_4KB</code>	4KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_8KB</code>	8KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_16KB</code>	16KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_32KB</code>	32KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_64KB</code>	64KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_128KB</code>	128KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_256KB</code>	256KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_512KB</code>	512KB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_1MB</code>	1MB Size of the MPU protection region
<code>LL_MPU_REGION_SIZE_2MB</code>	2MB Size of the MPU protection region

LL_MPU_REGION_SIZE_4MB	4MB Size of the MPU protection region
LL_MPU_REGION_SIZE_8MB	8MB Size of the MPU protection region
LL_MPU_REGION_SIZE_16MB	16MB Size of the MPU protection region
LL_MPU_REGION_SIZE_32MB	32MB Size of the MPU protection region
LL_MPU_REGION_SIZE_64MB	64MB Size of the MPU protection region
LL_MPU_REGION_SIZE_128MB	128MB Size of the MPU protection region
LL_MPU_REGION_SIZE_256MB	256MB Size of the MPU protection region
LL_MPU_REGION_SIZE_512MB	512MB Size of the MPU protection region
LL_MPU_REGION_SIZE_1GB	1GB Size of the MPU protection region
LL_MPU_REGION_SIZE_2GB	2GB Size of the MPU protection region
LL_MPU_REGION_SIZE_4GB	4GB Size of the MPU protection region

MPU Shareable Access

LL_MPU_ACCESS_SHAREABLE	Shareable memory attribute
LL_MPU_ACCESS_NOT_SHAREABLE	Not Shareable memory attribute

MPU TEX Level

LL_MPU_TEX_LEVEL0	b000 for TEX bits
LL_MPU_TEX_LEVEL1	b001 for TEX bits
LL_MPU_TEX_LEVEL2	b010 for TEX bits
LL_MPU_TEX_LEVEL4	b100 for TEX bits

52 LL CRC Generic Driver

52.1 CRC Firmware driver API description

52.1.1 Detailed description of functions

LL_CRC_ResetCRCCalculationUnit

Function name **__STATIC_INLINE void LL_CRC_ResetCRCCalculationUnit (CRC_TypeDef * CRCx)**

Function description Reset the CRC calculation unit.

Parameters • **CRCx:** CRC Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• CR RESET LL_CRC_ResetCRCCalculationUnit

LL_CRC_FeedData32

Function name **__STATIC_INLINE void LL_CRC_FeedData32 (CRC_TypeDef * CRCx, uint32_t InData)**

Function description Write given 32-bit data to the CRC calculator.

Parameters • **CRCx:** CRC Instance
• **InData:** value to be provided to CRC calculator between
between Min_Data=0 and Max_Data=0xFFFFFFFF

Return values • **None:**

Reference Manual to
LL API cross
reference:
• DR DR LL_CRC_FeedData32

LL_CRC_ReadData32

Function name **__STATIC_INLINE uint32_t LL_CRC_ReadData32 (CRC_TypeDef * CRCx)**

Function description Return current CRC calculation result.

Parameters • **CRCx:** CRC Instance

Return values • **Current:** CRC calculation result as stored in CRC_DR
register (32 bits).

Reference Manual to
LL API cross
reference:
• DR DR LL_CRC_ReadData32

LL_CRC_Read_IDR

Function name **__STATIC_INLINE uint32_t LL_CRC_Read_IDR (CRC_TypeDef**

*** CRCx)**

Function description	Return data stored in the Independent Data(IDR) register.
Parameters	<ul style="list-style-type: none"> • CRCx: CRC Instance
Return values	<ul style="list-style-type: none"> • Value: stored in CRC_IDR register (General-purpose 8-bit data register).
Notes	<ul style="list-style-type: none"> • This register can be used as a temporary storage location for one byte.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • IDR IDR LL_CRC_Read_IDR

LL_CRC_Write_IDR

Function name	<u>__STATIC_INLINE void LL_CRC_Write_IDR (CRC_TypeDef * CRCx, uint32_t InData)</u>
Function description	Store data in the Independent Data(IDR) register.
Parameters	<ul style="list-style-type: none"> • CRCx: CRC Instance • InData: value to be stored in CRC_IDR register (8-bit) between Min_Data=0 and Max_Data=0xFF
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This register can be used as a temporary storage location for one byte.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • IDR IDR LL_CRC_Write_IDR

LL_CRC_DeInit

Function name	<u>ErrorStatus LL_CRC_DeInit (CRC_TypeDef * CRCx)</u>
Function description	De-initialize CRC registers (Registers restored to their default values).
Parameters	<ul style="list-style-type: none"> • CRCx: CRC Instance
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: CRC registers are de-initialized – ERROR: CRC registers are not de-initialized

52.2 CRC Firmware driver defines**52.2.1 CRC*****Common Write and read registers Macros***

LL_CRC_WriteReg	Description:
	<ul style="list-style-type: none"> • Write a value in CRC register.
	Parameters:
	<ul style="list-style-type: none"> • <u>__INSTANCE__</u>: CRC Instance

- REG: Register to be written
- VALUE: Value to be written in the register

Return value:

- None

LL_CRC_ReadReg

- Read a value in CRC register.

Parameters:

- INSTANCE: CRC Instance
- REG: Register to be read

Return value:

- Register: value

53 LL DAC Generic Driver

53.1 DAC Firmware driver registers structures

53.1.1 LL_DAC_InitTypeDef

Data Fields

- *uint32_t TriggerSource*
- *uint32_t WaveAutoGeneration*
- *uint32_t WaveAutoGenerationConfig*
- *uint32_t OutputBuffer*

Field Documentation

- ***uint32_t LL_DAC_InitTypeDef::TriggerSource***
Set the conversion trigger source for the selected DAC channel: internal (SW start) or from external IP (timer event, external interrupt line). This parameter can be a value of **DAC_LL_EC_TRIGGER_SOURCE** This feature can be modified afterwards using unitary function **LL_DAC_SetTriggerSource()**.
- ***uint32_t LL_DAC_InitTypeDef::WaveAutoGeneration***
Set the waveform automatic generation mode for the selected DAC channel. This parameter can be a value of **DAC_LL_EC_WAVE_AUTO_GENERATION_MODE** This feature can be modified afterwards using unitary function **LL_DAC_SetWaveAutoGeneration()**.
- ***uint32_t LL_DAC_InitTypeDef::WaveAutoGenerationConfig***
Set the waveform automatic generation mode for the selected DAC channel. If waveform automatic generation mode is set to noise, this parameter can be a value of **DAC_LL_EC_WAVE_NOISE_LFSR_UNMASK_BITS** If waveform automatic generation mode is set to triangle, this parameter can be a value of **DAC_LL_EC_WAVE_TRIANGLE_AMPLITUDE**
Note:If waveform automatic generation mode is disabled, this parameter is discarded. This feature can be modified afterwards using unitary function **LL_DAC_SetWaveNoiseLFSR()** or **LL_DAC_SetWaveTriangleAmplitude()**, depending on the wave automatic generation selected.
- ***uint32_t LL_DAC_InitTypeDef::OutputBuffer***
Set the output buffer for the selected DAC channel. This parameter can be a value of **DAC_LL_EC_OUTPUT_BUFFER** This feature can be modified afterwards using unitary function **LL_DAC_SetOutputBuffer()**.

53.2 DAC Firmware driver API description

53.2.1 Detailed description of functions

LL_DAC_SetTriggerSource

Function name **_STATIC_INLINE void LL_DAC_SetTriggerSource
(DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t
TriggerSource)**

Function description Set the conversion trigger source for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following

	<p>values:</p> <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
	<ul style="list-style-type: none"> • TriggerSource: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_TRIG_SOFTWARE - LL_DAC_TRIG_EXT_TIM2_TRGO - LL_DAC_TRIG_EXT_TIM4_TRGO - LL_DAC_TRIG_EXT_TIM6_TRGO - LL_DAC_TRIG_EXT_TIM7_TRGO - LL_DAC_TRIG_EXT_TIM9_TRGO - LL_DAC_TRIG_EXT EXTI_LINE9
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • For conversion trigger source to be effective, DAC trigger must be enabled using function <code>LL_DAC_EnableTrigger()</code>. • To set conversion trigger source, DAC channel must be disabled. Otherwise, the setting is discarded. • Availability of parameters of trigger sources from timer depends on timers availability on the selected device.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TSEL1 <code>LL_DAC_SetTriggerSource</code> • CR TSEL2 <code>LL_DAC_SetTriggerSource</code>

LL_DAC_GetTriggerSource

Function name	<code>_STATIC_INLINE uint32_t LL_DAC_GetTriggerSource (DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Get the conversion trigger source for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_TRIG_SOFTWARE - LL_DAC_TRIG_EXT_TIM2_TRGO - LL_DAC_TRIG_EXT_TIM4_TRGO - LL_DAC_TRIG_EXT_TIM6_TRGO - LL_DAC_TRIG_EXT_TIM7_TRGO - LL_DAC_TRIG_EXT_TIM9_TRGO - LL_DAC_TRIG_EXT EXTI_LINE9
Notes	<ul style="list-style-type: none"> • For conversion trigger source to be effective, DAC trigger must be enabled using function <code>LL_DAC_EnableTrigger()</code>. • Availability of parameters of trigger sources from timer depends on timers availability on the selected device.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TSEL1 <code>LL_DAC_GetTriggerSource</code> • CR TSEL2 <code>LL_DAC_GetTriggerSource</code>

LL_DAC_SetWaveAutoGeneration

Function name	<code>__STATIC_INLINE void LL_DAC_SetWaveAutoGeneration(DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t WaveAutoGeneration)</code>
Function description	Set the waveform automatic generation mode for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2 • WaveAutoGeneration: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_WAVE_AUTO_GENERATION_NONE - LL_DAC_WAVE_AUTO_GENERATION_NOISE - LL_DAC_WAVE_AUTO_GENERATION_TRIANGLE
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR WAVE1 LL_DAC_SetWaveAutoGeneration • CR WAVE2 LL_DAC_SetWaveAutoGeneration

LL_DAC_GetWaveAutoGeneration

Function name	<code>__STATIC_INLINE uint32_t LL_DAC_GetWaveAutoGeneration(DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Get the waveform automatic generation mode for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_WAVE_AUTO_GENERATION_NONE - LL_DAC_WAVE_AUTO_GENERATION_NOISE - LL_DAC_WAVE_AUTO_GENERATION_TRIANGLE
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR WAVE1 LL_DAC_GetWaveAutoGeneration • CR WAVE2 LL_DAC_GetWaveAutoGeneration

LL_DAC_SetWaveNoiseLFSR

Function name	<code>__STATIC_INLINE void LL_DAC_SetWaveNoiseLFSR(DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t NoiseLFSRMask)</code>
Function description	Set the noise waveform generation for the selected DAC channel: Noise mode and parameters LFSR (linear feedback shift register).
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance

- **DAC_Channel:** This parameter can be one of the following values:
 - LL_DAC_CHANNEL_1
 - LL_DAC_CHANNEL_2
- **NoiseLFSRMask:** This parameter can be one of the following values:
 - LL_DAC_NOISE_LFSR_UNMASK_BIT0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS1_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS2_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS3_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS4_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS5_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS6_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS7_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS8_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS9_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS10_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS11_0

Return values

- **None:**

Notes

- For wave generation to be effective, DAC channel wave generation mode must be enabled using function `LL_DAC_SetWaveAutoGeneration()`.
- This setting can be set when the selected DAC channel is disabled (otherwise, the setting operation is ignored).

Reference Manual to
LL API cross
reference:

- CR MAMP1 `LL_DAC_SetWaveNoiseLFSR`
- CR MAMP2 `LL_DAC_SetWaveNoiseLFSR`

LL_DAC_GetWaveNoiseLFSR

Function name

```
STATIC_INLINE uint32_t LL_DAC_GetWaveNoiseLFSR
(DAC_TypeDef * DACx, uint32_t DAC_Channel)
```

Function description

Set the noise waveform generation for the selected DAC channel:
Noise mode and parameters LFSR (linear feedback shift register).

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
 - LL_DAC_CHANNEL_1
 - LL_DAC_CHANNEL_2

Return values

- **Returned:** value can be one of the following values:
 - LL_DAC_NOISE_LFSR_UNMASK_BIT0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS1_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS2_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS3_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS4_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS5_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS6_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS7_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS8_0
 - LL_DAC_NOISE_LFSR_UNMASK_BITS9_0

Reference Manual to
LL API cross
reference:

- LL_DAC_NOISE_LFSR_UNMASK_BITS10_0
- LL_DAC_NOISE_LFSR_UNMASK_BITS11_0

- CR MAMP1 LL_DAC_GetWaveNoiseLFSR
- CR MAMP2 LL_DAC_GetWaveNoiseLFSR

LL_DAC_SetWaveTriangleAmplitude

Function name **`__STATIC_INLINE void LL_DAC_SetWaveTriangleAmplitude(DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t TriangleAmplitude)`**

Function description Set the triangle waveform generation for the selected DAC channel: triangle mode and amplitude.

- Parameters
- **DACx:** DAC instance
 - **DAC_Channel:** This parameter can be one of the following values:
 - LL_DAC_CHANNEL_1
 - LL_DAC_CHANNEL_2
 - **TriangleAmplitude:** This parameter can be one of the following values:
 - LL_DAC_TRIANGLE_AMPLITUDE_1
 - LL_DAC_TRIANGLE_AMPLITUDE_3
 - LL_DAC_TRIANGLE_AMPLITUDE_7
 - LL_DAC_TRIANGLE_AMPLITUDE_15
 - LL_DAC_TRIANGLE_AMPLITUDE_31
 - LL_DAC_TRIANGLE_AMPLITUDE_63
 - LL_DAC_TRIANGLE_AMPLITUDE_127
 - LL_DAC_TRIANGLE_AMPLITUDE_255
 - LL_DAC_TRIANGLE_AMPLITUDE_511
 - LL_DAC_TRIANGLE_AMPLITUDE_1023
 - LL_DAC_TRIANGLE_AMPLITUDE_2047
 - LL_DAC_TRIANGLE_AMPLITUDE_4095

- Return values
- **None:**

- Notes
- For wave generation to be effective, DAC channel wave generation mode must be enabled using function `LL_DAC_SetWaveAutoGeneration()`.
 - This setting can be set when the selected DAC channel is disabled (otherwise, the setting operation is ignored).

Reference Manual to
LL API cross
reference:

- CR MAMP1 LL_DAC_SetWaveTriangleAmplitude
- CR MAMP2 LL_DAC_SetWaveTriangleAmplitude

LL_DAC_GetWaveTriangleAmplitude

Function name **`__STATIC_INLINE uint32_t LL_DAC_GetWaveTriangleAmplitude (DAC_TypeDef * DACx, uint32_t DAC_Channel)`**

Function description Set the triangle waveform generation for the selected DAC channel: triangle mode and amplitude.

Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DAC_CHANNEL_1 – LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_DAC_TRIANGLE_AMPLITUDE_1 – LL_DAC_TRIANGLE_AMPLITUDE_3 – LL_DAC_TRIANGLE_AMPLITUDE_7 – LL_DAC_TRIANGLE_AMPLITUDE_15 – LL_DAC_TRIANGLE_AMPLITUDE_31 – LL_DAC_TRIANGLE_AMPLITUDE_63 – LL_DAC_TRIANGLE_AMPLITUDE_127 – LL_DAC_TRIANGLE_AMPLITUDE_255 – LL_DAC_TRIANGLE_AMPLITUDE_511 – LL_DAC_TRIANGLE_AMPLITUDE_1023 – LL_DAC_TRIANGLE_AMPLITUDE_2047 – LL_DAC_TRIANGLE_AMPLITUDE_4095
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR MAMP1 LL_DAC_GetWaveTriangleAmplitude • CR MAMP2 LL_DAC_GetWaveTriangleAmplitude

LL_DAC_SetOutputBuffer

Function name	<code>__STATIC_INLINE void LL_DAC_SetOutputBuffer (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t OutputBuffer)</code>
Function description	Set the output buffer for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DAC_CHANNEL_1 – LL_DAC_CHANNEL_2 • OutputBuffer: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DAC_OUTPUT_BUFFER_ENABLE – LL_DAC_OUTPUT_BUFFER_DISABLE
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR BOFF1 LL_DAC_SetOutputBuffer • CR BOFF2 LL_DAC_SetOutputBuffer

LL_DAC_GetOutputBuffer

Function name	<code>__STATIC_INLINE uint32_t LL_DAC_GetOutputBuffer (DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Get the output buffer state for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following

	<p>values:</p> <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_OUTPUT_BUFFER_ENABLE - LL_DAC_OUTPUT_BUFFER_DISABLE
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR BOFF1 LL_DAC_GetOutputBuffer • CR BOFF2 LL_DAC_GetOutputBuffer
LL_DAC_EnableDMAReq	
Function name	<code>_STATIC_INLINE void LL_DAC_EnableDMAReq(DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Enable DAC DMA transfer request of the selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • To configure DMA source address (peripheral address), use function LL_DAC_DMA_GetRegAddr().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR DMAEN1 LL_DAC_EnableDMAReq • CR DMAEN2 LL_DAC_EnableDMAReq
LL_DAC_DisableDMAReq	
Function name	<code>_STATIC_INLINE void LL_DAC_DisableDMAReq(DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Disable DAC DMA transfer request of the selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • To configure DMA source address (peripheral address), use function LL_DAC_DMA_GetRegAddr().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR DMAEN1 LL_DAC_DisableDMAReq • CR DMAEN2 LL_DAC_DisableDMAReq

LL_DAC_IsDMAReqEnabled

Function name	<code>_STATIC_INLINE uint32_t LL_DAC_IsDMAReqEnabled</code>
---------------	--

(DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description	Get DAC DMA transfer request state of the selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR DMAEN1 LL_DAC_IsDMAReqEnabled • CR DMAEN2 LL_DAC_IsDMAReqEnabled

LL_DAC_DMA_GetRegAddr

Function name	__STATIC_INLINE uint32_t LL_DAC_DMA_GetRegAddr (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Register)
Function description	Function to help to configure DMA transfer to DAC: retrieve the DAC register address from DAC instance and a list of DAC registers intended to be used (most commonly) with DMA transfer.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2 • Register: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_DMA_REG_DATA_12BITS_RIGHT_ALIGNED - LL_DAC_DMA_REG_DATA_12BITS_LEFT_ALIGNED - LL_DAC_DMA_REG_DATA_8BITS_RIGHT_ALIGNED
Return values	<ul style="list-style-type: none"> • DAC: register address
Notes	<ul style="list-style-type: none"> • These DAC registers are data holding registers: when DAC conversion is requested, DAC generates a DMA transfer request to have data available in DAC data holding registers. • This macro is intended to be used with LL DMA driver, refer to function "LL_DMA_ConfigAddresses()". Example: <pre>LL_DMA_ConfigAddresses(DMA1, LL_DMA_CHANNEL_1, (uint32_t)&< array or variable >, LL_DAC_DMA_GetRegAddr(DAC1, LL_DAC_CHANNEL_1, LL_DAC_DMA_REG_DATA_12BITS_RIGHT_ALIGNED), LL_DMA_DIRECTION_MEMORY_TO_PERIPH);</pre>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DHR12R1 DACC1DHR LL_DAC_DMA_GetRegAddr • DHR12L1 DACC1DHR LL_DAC_DMA_GetRegAddr • DHR8R1 DACC1DHR LL_DAC_DMA_GetRegAddr • DHR12R2 DACC2DHR LL_DAC_DMA_GetRegAddr • DHR12L2 DACC2DHR LL_DAC_DMA_GetRegAddr • DHR8R2 DACC2DHR LL_DAC_DMA_GetRegAddr

LL_DAC_Enable

Function name	__STATIC_INLINE void LL_DAC_Enable (DAC_TypeDef *
---------------	--

DACx, uint32_t DAC_Channel)

Function description	Enable DAC selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • After enable from off state, DAC channel requires a delay for output voltage to reach accuracy +/- 1 LSB. Refer to device datasheet, parameter "tWAKEUP".
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR EN1 LL_DAC_Enable • CR EN2 LL_DAC_Enable

LL_DAC_Disable

Function name	_STATIC_INLINE void LL_DAC_Disable (DAC_TypeDef * DACx, uint32_t DAC_Channel)
Function description	Disable DAC selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR EN1 LL_DAC_Disable • CR EN2 LL_DAC_Disable

LL_DAC_IsEnabled

Function name	_STATIC_INLINE uint32_t LL_DAC_IsEnabled (DAC_TypeDef * DACx, uint32_t DAC_Channel)
Function description	Get DAC enable state of the selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR EN1 LL_DAC_IsEnabled • CR EN2 LL_DAC_IsEnabled

LL_DAC_EnableTrigger

Function name	<code>__STATIC_INLINE void LL_DAC_EnableTrigger (DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Enable DAC trigger of the selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • - If DAC trigger is disabled, DAC conversion is performed automatically once the data holding register is updated, using functions "LL_DAC_ConvertData{8; 12}{Right; Left}Aligned()": LL_DAC_ConvertData12RightAligned(), ... If DAC trigger is enabled, DAC conversion is performed only when a hardware or software trigger event is occurring. Select trigger source using function LL_DAC_SetTriggerSource().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TEN1 LL_DAC_EnableTrigger • CR TEN2 LL_DAC_EnableTrigger

LL_DAC_DisableTrigger

Function name	<code>__STATIC_INLINE void LL_DAC_DisableTrigger (DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Disable DAC trigger of the selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TEN1 LL_DAC_DisableTrigger • CR TEN2 LL_DAC_DisableTrigger

LL_DAC_IsTriggerEnabled

Function name	<code>__STATIC_INLINE uint32_t LL_DAC_IsTriggerEnabled (DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Get DAC trigger state of the selected channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- CR TEN1 LL_DAC_IsTriggerEnabled
 - CR TEN2 LL_DAC_IsTriggerEnabled

LL_DAC_TrigSWConversion

Function name	<code>_STATIC_INLINE void LL_DAC_TrigSWConversion (DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Trig DAC conversion by software for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can a combination of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Preliminarily, DAC trigger must be set to software trigger using function LL_DAC_SetTriggerSource() with parameter "LL_DAC_TRIGGER_SOFTWARE". and DAC trigger must be enabled using function LL_DAC_EnableTrigger(). • For devices featuring DAC with 2 channels: this function can perform a SW start of both DAC channels simultaneously. Two channels can be selected as parameter. Example: (LL_DAC_CHANNEL_1 LL_DAC_CHANNEL_2)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SWTRIGR SWTRIG1 LL_DAC_TrigSWConversion • SWTRIGR SWTRIG2 LL_DAC_TrigSWConversion

LL_DAC_ConvertData12RightAligned

Function name	<code>_STATIC_INLINE void LL_DAC_ConvertData12RightAligned (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)</code>
Function description	Set the data to be loaded in the data holding register in format 12 bits left alignment (LSB aligned on bit 0), for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2 • Data: Value between Min_Data=0x000 and Max_Data=0xFFFF
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DHR12R1 DACC1DHR LL_DAC_ConvertData12RightAligned • DHR12R2 DACC2DHR LL_DAC_ConvertData12RightAligned

LL_DAC_ConvertData12LeftAligned

Function name	<code>_STATIC_INLINE void LL_DAC_ConvertData12LeftAligned</code>
---------------	--

	(DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)
Function description	Set the data to be loaded in the data holding register in format 12 bits left alignment (MSB aligned on bit 15), for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2 • Data: Value between Min_Data=0x000 and Max_Data=0xFFFF
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DHR12L1 DACC1DHR LL_DAC_ConvertData12LeftAligned • DHR12L2 DACC2DHR LL_DAC_ConvertData12LeftAligned

LL_DAC_ConvertData8RightAligned

Function name	_STATIC_INLINE void LL_DAC_ConvertData8RightAligned (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)
Function description	Set the data to be loaded in the data holding register in format 8 bits left alignment (LSB aligned on bit 0), for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2 • Data: Value between Min_Data=0x00 and Max_Data=0xFF
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DHR8R1 DACC1DHR LL_DAC_ConvertData8RightAligned • DHR8R2 DACC2DHR LL_DAC_ConvertData8RightAligned

LL_DAC_ConvertDualData12RightAligned

Function name	_STATIC_INLINE void LL_DAC_ConvertDualData12RightAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)
Function description	Set the data to be loaded in the data holding register in format 12 bits left alignment (LSB aligned on bit 0), for both DAC channels.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DataChannel1: Value between Min_Data=0x000 and Max_Data=0xFFFF • DataChannel2: Value between Min_Data=0x000 and Max_Data=0xFFFF
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to	<ul style="list-style-type: none"> • DHR12RD DACC1DHR

LL API cross reference:	<ul style="list-style-type: none"> LL_DAC_ConvertDualData12RightAligned DHR12RD DACC2DHR LL_DAC_ConvertDualData12RightAligned
-------------------------	--

LL_DAC_ConvertDualData12LeftAligned

Function name	<code>__STATIC_INLINE void LL_DAC_ConvertDualData12LeftAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)</code>
Function description	Set the data to be loaded in the data holding register in format 12 bits left alignment (MSB aligned on bit 15), for both DAC channels.
Parameters	<ul style="list-style-type: none"> DACx: DAC instance DataChannel1: Value between Min_Data=0x000 and Max_Data=0xFFFF DataChannel2: Value between Min_Data=0x000 and Max_Data=0xFFFF
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DHR12LD DACC1DHR LL_DAC_ConvertDualData12LeftAligned DHR12LD DACC2DHR LL_DAC_ConvertDualData12LeftAligned

LL_DAC_ConvertDualData8RightAligned

Function name	<code>__STATIC_INLINE void LL_DAC_ConvertDualData8RightAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)</code>
Function description	Set the data to be loaded in the data holding register in format 8 bits left alignment (LSB aligned on bit 0), for both DAC channels.
Parameters	<ul style="list-style-type: none"> DACx: DAC instance DataChannel1: Value between Min_Data=0x00 and Max_Data=0xFF DataChannel2: Value between Min_Data=0x00 and Max_Data=0xFF
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DHR8RD DACC1DHR LL_DAC_ConvertDualData8RightAligned DHR8RD DACC2DHR LL_DAC_ConvertDualData8RightAligned

LL_DAC_RetrieveOutputData

Function name	<code>__STATIC_INLINE uint32_t LL_DAC_RetrieveOutputData (DAC_TypeDef * DACx, uint32_t DAC_Channel)</code>
Function description	Retrieve output data currently generated for the selected DAC channel.
Parameters	<ul style="list-style-type: none"> DACx: DAC instance DAC_Channel: This parameter can be one of the following values:

	<ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x000 and Max_Data=0xFFFF
Notes	<ul style="list-style-type: none"> • Whatever alignment and resolution settings (using functions "LL_DAC_ConvertData{8; 12}{Right; Left} Aligned()": LL_DAC_ConvertData12RightAligned(), ...), output data format is 12 bits right aligned (LSB aligned on bit 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DOR1 DACC1DOR LL_DAC_RetrieveOutputData • DOR2 DACC2DOR LL_DAC_RetrieveOutputData

LL_DAC_IsActiveFlag_DMAUDR1

Function name	<code>__STATIC_INLINE uint32_t LL_DAC_IsActiveFlag_DMAUDR1(DAC_TypeDef * DACx)</code>
Function description	Get DAC underrun flag for DAC channel 1.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR DMAUDR1 LL_DAC_IsActiveFlag_DMAUDR1

LL_DAC_IsActiveFlag_DMAUDR2

Function name	<code>__STATIC_INLINE uint32_t LL_DAC_IsActiveFlag_DMAUDR2(DAC_TypeDef * DACx)</code>
Function description	Get DAC underrun flag for DAC channel 2.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR DMAUDR2 LL_DAC_IsActiveFlag_DMAUDR2

LL_DAC_ClearFlag_DMAUDR1

Function name	<code>__STATIC_INLINE void LL_DAC_ClearFlag_DMAUDR1(DAC_TypeDef * DACx)</code>
Function description	Clear DAC underrun flag for DAC channel 1.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR DMAUDR1 LL_DAC_ClearFlag_DMAUDR1

LL_DAC_ClearFlag_DMAUDR2

Function name	<code>__STATIC_INLINE void LL_DAC_ClearFlag_DMAUDR2(DAC_TypeDef * DACx)</code>
Function description	Clear DAC underrun flag for DAC channel 2.
Parameters	<ul style="list-style-type: none">• DACx: DAC instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• SR DMAUDR2 LL_DAC_ClearFlag_DMAUDR2

LL_DAC_EnableIT_DMAUDR1

Function name	<code>__STATIC_INLINE void LL_DAC_EnableIT_DMAUDR1(DAC_TypeDef * DACx)</code>
Function description	Enable DMA underrun interrupt for DAC channel 1.
Parameters	<ul style="list-style-type: none">• DACx: DAC instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR DMAUDRIE1 LL_DAC_EnableIT_DMAUDR1

LL_DAC_EnableIT_DMAUDR2

Function name	<code>__STATIC_INLINE void LL_DAC_EnableIT_DMAUDR2(DAC_TypeDef * DACx)</code>
Function description	Enable DMA underrun interrupt for DAC channel 2.
Parameters	<ul style="list-style-type: none">• DACx: DAC instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR DMAUDRIE2 LL_DAC_EnableIT_DMAUDR2

LL_DAC_DisableIT_DMAUDR1

Function name	<code>__STATIC_INLINE void LL_DAC_DisableIT_DMAUDR1(DAC_TypeDef * DACx)</code>
Function description	Disable DMA underrun interrupt for DAC channel 1.
Parameters	<ul style="list-style-type: none">• DACx: DAC instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR DMAUDRIE1 LL_DAC_DisableIT_DMAUDR1

LL_DAC_DisableIT_DMAUDR2

Function name	<code>__STATIC_INLINE void LL_DAC_DisableIT_DMAUDR2 (DAC_TypeDef * DACx)</code>
Function description	Disable DMA underrun interrupt for DAC channel 2.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR DMAUDRIE2 LL_DAC_DisableIT_DMAUDR2

LL_DAC_IsEnabledIT_DMAUDR1

Function name	<code>__STATIC_INLINE uint32_t LL_DAC_IsEnabledIT_DMAUDR1 (DAC_TypeDef * DACx)</code>
Function description	Get DMA underrun interrupt for DAC channel 1.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR DMAUDRIE1 LL_DAC_IsEnabledIT_DMAUDR1

LL_DAC_IsEnabledIT_DMAUDR2

Function name	<code>__STATIC_INLINE uint32_t LL_DAC_IsEnabledIT_DMAUDR2 (DAC_TypeDef * DACx)</code>
Function description	Get DMA underrun interrupt for DAC channel 2.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR DMAUDRIE2 LL_DAC_IsEnabledIT_DMAUDR2

LL_DAC_DeInit

Function name	ErrorStatus LL_DAC_DeInit (DAC_TypeDef * DACx)
Function description	De-initialize registers of the selected DAC instance to their default reset values.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: DAC registers are de-initialized – ERROR: not applicable

LL_DAC_Init

Function name	ErrorStatus LL_DAC_Init (DAC_TypeDef * DACx, uint32_t
---------------	--

DAC_Channel, LL_DAC_InitTypeDef * DAC_InitStruct)

Function description	Initialize some features of DAC instance.
Parameters	<ul style="list-style-type: none"> • DACx: DAC instance • DAC_Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DAC_CHANNEL_1 - LL_DAC_CHANNEL_2 • DAC_InitStruct: Pointer to a LL_DAC_InitTypeDef structure
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> - SUCCESS: DAC registers are initialized - ERROR: DAC registers are not initialized
Notes	<ul style="list-style-type: none"> • The setting of these parameters by function LL_DAC_Init() is conditioned to DAC state: DAC instance must be disabled.

LL_DAC_StructInit

Function name	void LL_DAC_StructInit (LL_DAC_InitTypeDef * DAC_InitStruct)
Function description	Set each LL_DAC_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • DAC_InitStruct: pointer to a LL_DAC_InitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> • None:

53.3 DAC Firmware driver defines

53.3.1 DAC

DAC channels

LL_DAC_CHANNEL_1 DAC channel 1

LL_DAC_CHANNEL_2 DAC channel 2

DAC flags

LL_DAC_FLAG_DMAUDR1 DAC channel 1 flag DMA underrun

LL_DAC_FLAG_DMAUDR2 DAC channel 2 flag DMA underrun

Definitions of DAC hardware constraints delays

LL_DAC_DELAY_STARTUP_VOLTAGE_SETTLING_US Delay for DAC channel voltage settling time from DAC channel startup (transition from disable to enable)

LL_DAC_DELAY_VOLTAGE_SETTLING_US Delay for DAC channel voltage settling time

DAC interruptions

LL_DAC_IT_DMAUDRIE1 DAC channel 1 interruption DMA underrun

LL_DAC_IT_DMAUDRIE2 DAC channel 2 interruption DMA underrun

DAC channel output buffer

`LL_DAC_OUTPUT_BUFFER_ENABLE` The selected DAC channel output is buffered: higher drive current capability, but also higher current consumption

`LL_DAC_OUTPUT_BUFFER_DISABLE` The selected DAC channel output is not buffered: lower drive current capability, but also lower current consumption

DAC registers compliant with specific purpose

`LL_DAC_DMA_REG_DATA_12BITS_RIGHT_ALIGNED` DAC channel data holding register 12 bits right aligned

`LL_DAC_DMA_REG_DATA_12BITS_LEFT_ALIGNED` DAC channel data holding register 12 bits left aligned

`LL_DAC_DMA_REG_DATA_8BITS_RIGHT_ALIGNED` DAC channel data holding register 8 bits right aligned

DAC channel output resolution

`LL_DAC_RESOLUTION_12B` DAC channel resolution 12 bits

`LL_DAC_RESOLUTION_8B` DAC channel resolution 8 bits

DAC trigger source

`LL_DAC_TRIG_SOFTWARE` DAC channel conversion trigger internal (SW start)

`LL_DAC_TRIG_EXT_TIM2_TRGO` DAC channel conversion trigger from external IP: TIM2 TRGO.

`LL_DAC_TRIG_EXT_TIM4_TRGO` DAC channel conversion trigger from external IP: TIM4 TRGO.

`LL_DAC_TRIG_EXT_TIM6_TRGO` DAC channel conversion trigger from external IP: TIM6 TRGO.

`LL_DAC_TRIG_EXT_TIM7_TRGO` DAC channel conversion trigger from external IP: TIM7 TRGO.

`LL_DAC_TRIG_EXT_TIM9_TRGO` DAC channel conversion trigger from external IP: TIM15 TRGO.

`LL_DAC_TRIG_EXT EXTI_LINE9` DAC channel conversion trigger from external IP: external interrupt line 9.

DAC waveform automatic generation mode

`LL_DAC_WAVE_AUTO_GENERATION_NONE` DAC channel wave auto generation mode disabled.

`LL_DAC_WAVE_AUTO_GENERATION_NOISE` DAC channel wave auto generation mode enabled, set generated noise waveform.

`LL_DAC_WAVE_AUTO_GENERATION_TRIANGLE` DAC channel wave auto generation mode enabled, set generated triangle waveform.

DAC wave generation - Noise LFSR unmask bits

`LL_DAC_NOISE_LFSR_UNMASK_BIT0` Noise wave generation, unmask LFSR bit0, for the selected DAC channel

`LL_DAC_NOISE_LFSR_UNMASK_BITS1_0` Noise wave generation, unmask LFSR

	bits[1:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS2_0	Noise wave generation, unmask LFSR bits[2:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS3_0	Noise wave generation, unmask LFSR bits[3:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS4_0	Noise wave generation, unmask LFSR bits[4:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS5_0	Noise wave generation, unmask LFSR bits[5:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS6_0	Noise wave generation, unmask LFSR bits[6:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS7_0	Noise wave generation, unmask LFSR bits[7:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS8_0	Noise wave generation, unmask LFSR bits[8:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS9_0	Noise wave generation, unmask LFSR bits[9:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS10_0	Noise wave generation, unmask LFSR bits[10:0], for the selected DAC channel
LL_DAC_NOISE_LFSR_UNMASK_BITS11_0	Noise wave generation, unmask LFSR bits[11:0], for the selected DAC channel

DAC wave generation - Triangle amplitude

LL_DAC_TRIANGLE_AMPLITUDE_1	Triangle wave generation, amplitude of 1 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_3	Triangle wave generation, amplitude of 3 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_7	Triangle wave generation, amplitude of 7 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_15	Triangle wave generation, amplitude of 15 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_31	Triangle wave generation, amplitude of 31 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_63	Triangle wave generation, amplitude of 63 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_127	Triangle wave generation, amplitude of 127 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_255	Triangle wave generation, amplitude of 255 LSB of DAC output range, for the selected

	DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_511	Triangle wave generation, amplitude of 512 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_1023	Triangle wave generation, amplitude of 1023 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_2047	Triangle wave generation, amplitude of 2047 LSB of DAC output range, for the selected DAC channel
LL_DAC_TRIANGLE_AMPLITUDE_4095	Triangle wave generation, amplitude of 4095 LSB of DAC output range, for the selected DAC channel

DAC helper macro

`_LL_DAC_CHANNEL_TO_DECIMAL_
NB`

Description:

- Helper macro to get DAC channel number in decimal format from literals `LL_DAC_CHANNEL_x`.

Parameters:

- `_CHANNEL_`: This parameter can be one of the following values:
 - `LL_DAC_CHANNEL_1`
 - `LL_DAC_CHANNEL_2`

Return value:

- 1...2

Notes:

- The input can be a value from functions where a channel number is returned.

`_LL_DAC_DECIMAL_NB_TO_CHANN
EL`

Description:

- Helper macro to get DAC channel in literal format `LL_DAC_CHANNEL_x` from number in decimal format.

Parameters:

- `_DECIMAL_NB_`: 1...2

Return value:

- Returned: value can be one of the following values:
 - `LL_DAC_CHANNEL_1`
 - `LL_DAC_CHANNEL_2`

Notes:

- If the input parameter does not correspond to a DAC channel, this macro returns value '0'.

[__LL_DAC_DIGITAL_SCALE](#)**Description:**

- Helper macro to define the DAC conversion data full-scale digital value corresponding to the selected DAC resolution.

Parameters:

- [__DAC_RESOLUTION__](#): This parameter can be one of the following values:
 - [LL_DAC_RESOLUTION_12B](#)
 - [LL_DAC_RESOLUTION_8B](#)

Return value:

- ADC: conversion data equivalent voltage value (unit: mVolt)

Notes:

- DAC conversion data full-scale corresponds to voltage range determined by analog voltage references Vref+ and Vref- (refer to reference manual).

[__LL_DAC_CALC_VOLTAGE_TO_DAT_A](#)**Description:**

- Helper macro to calculate the DAC conversion data (unit: digital value) corresponding to a voltage (unit: mVolt).

Parameters:

- [__VREFANALOG_VOLTAGE__](#): Analog reference voltage (unit: mV)
- [__DAC_VOLTAGE__](#): Voltage to be generated by DAC channel (unit: mVolt).
- [__DAC_RESOLUTION__](#): This parameter can be one of the following values:
 - [LL_DAC_RESOLUTION_12B](#)
 - [LL_DAC_RESOLUTION_8B](#)

Return value:

- DAC: conversion data (unit: digital value)

Notes:

- This helper macro is intended to provide input data in voltage rather than digital value, to be used with LL DAC functions such as [LL_DAC_ConvertData12RightAligned\(\)](#). Analog reference voltage (Vref+) must be either known from user board environment or can be calculated using ADC measurement and ADC helper macro [__LL_ADC_CALC_VREFANALOG_VOLTAGE\(\)](#).

Common write and read registers macros**LL_DAC_WriteReg****Description:**

- Write a value in DAC register.

Parameters:

- __INSTANCE__: DAC Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_DAC_ReadReg**Description:**

- Read a value in DAC register.

Parameters:

- __INSTANCE__: DAC Instance
- __REG__: Register to be read

Return value:

- Register: value

54 LL DMA Generic Driver

54.1 DMA Firmware driver registers structures

54.1.1 LL_DMA_InitTypeDef

Data Fields

- *uint32_t PeriphOrM2MSrcAddress*
- *uint32_t MemoryOrM2MDstAddress*
- *uint32_t Direction*
- *uint32_t Mode*
- *uint32_t PeriphOrM2MSrcIncMode*
- *uint32_t MemoryOrM2MDstIncMode*
- *uint32_t PeriphOrM2MSrcDataSize*
- *uint32_t MemoryOrM2MDstDataSize*
- *uint32_t NbData*
- *uint32_t Priority*

Field Documentation

- ***uint32_t LL_DMA_InitTypeDef::PeriphOrM2MSrcAddress***
Specifies the peripheral base address for DMA transfer or as Source base address in case of memory to memory transfer direction. This parameter must be a value between Min_Data = 0 and Max_Data = 0xFFFFFFFF.
- ***uint32_t LL_DMA_InitTypeDef::MemoryOrM2MDstAddress***
Specifies the memory base address for DMA transfer or as Destination base address in case of memory to memory transfer direction. This parameter must be a value between Min_Data = 0 and Max_Data = 0xFFFFFFFF.
- ***uint32_t LL_DMA_InitTypeDef::Direction***
Specifies if the data will be transferred from memory to peripheral, from memory to memory or from peripheral to memory. This parameter can be a value of **DMA_LL_EC_DIRECTION** This feature can be modified afterwards using unitary function **LL_DMA_SetDataTransferDirection()**.
- ***uint32_t LL_DMA_InitTypeDef::Mode***
Specifies the normal or circular operation mode. This parameter can be a value of **DMA_LL_EC_MODE**
Note: The circular buffer mode cannot be used if the memory to memory data transfer direction is configured on the selected Channel This feature can be modified afterwards using unitary function **LL_DMA_SetMode()**.
- ***uint32_t LL_DMA_InitTypeDef::PeriphOrM2MSrcIncMode***
Specifies whether the Peripheral address or Source address in case of memory to memory transfer direction is incremented or not. This parameter can be a value of **DMA_LL_EC_PERIPH** This feature can be modified afterwards using unitary function **LL_DMA_SetPeriphIncMode()**.
- ***uint32_t LL_DMA_InitTypeDef::MemoryOrM2MDstIncMode***
Specifies whether the Memory address or Destination address in case of memory to memory transfer direction is incremented or not. This parameter can be a value of **DMA_LL_EC_MEMORY** This feature can be modified afterwards using unitary function **LL_DMA_SetMemoryIncMode()**.
- ***uint32_t LL_DMA_InitTypeDef::PeriphOrM2MSrcDataSize***
Specifies the Peripheral data size alignment or Source data size alignment (byte, half word, word) in case of memory to memory transfer direction. This parameter can be a

- value of **DMA_LL_EC_PDATAALIGN**This feature can be modified afterwards using unitary function **LL_DMA_SetPeriphSize()**.
- ***uint32_t LL_DMA_InitTypeDef::MemoryOrM2MDstDataSize***
Specifies the Memory data size alignment or Destination data size alignment (byte, half word, word) in case of memory to memory transfer direction. This parameter can be a value of **DMA_LL_EC_MDATAALIGN**This feature can be modified afterwards using unitary function **LL_DMA_SetMemorySize()**.
 - ***uint32_t LL_DMA_InitTypeDef::NbData***
Specifies the number of data to transfer, in data unit. The data unit is equal to the source buffer configuration set in PeripheralSize or MemorySize parameters depending in the transfer direction. This parameter must be a value between Min_Data = 0 and Max_Data = 0x0000FFFFThis feature can be modified afterwards using unitary function **LL_DMA_SetDataLength()**.
 - ***uint32_t LL_DMA_InitTypeDef::Priority***
Specifies the channel priority level. This parameter can be a value of **DMA_LL_EC_PRIORITY**This feature can be modified afterwards using unitary function **LL_DMA_SetChannelPriorityLevel()**.

54.2 DMA Firmware driver API description

54.2.1 Detailed description of functions

LL_DMA_EnableChannel

Function name	<code>__STATIC_INLINE void LL_DMA_EnableChannel (DMA_TypeDef * DMax, uint32_t Channel)</code>
Function description	Enable DMA channel.
Parameters	<ul style="list-style-type: none"> • DMax: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR EN LL_DMA_EnableChannel

LL_DMA_DisableChannel

Function name	<code>__STATIC_INLINE void LL_DMA_DisableChannel (DMA_TypeDef * DMax, uint32_t Channel)</code>
Function description	Disable DMA channel.
Parameters	<ul style="list-style-type: none"> • DMax: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3

	<ul style="list-style-type: none"> - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR EN LL_DMA_DisableChannel

LL_DMA_IsEnabledChannel

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsEnabledChannel(DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Check if DMA channel is enabled or disabled.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR EN LL_DMA_IsEnabledChannel

LL_DMA_ConfigTransfer

Function name	<code>__STATIC_INLINE void LL_DMA_ConfigTransfer(DMA_TypeDef * DMAx, uint32_t Channel, uint32_t Configuration)</code>
Function description	Configure all parameters link to DMA transfer.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7 • Configuration: This parameter must be a combination of all the following values: <ul style="list-style-type: none"> - LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH or LL_DMA_DIRECTION_MEMORY_TO_MEMORY - LL_DMA_MODE_NORMAL or

- LL_DMA_MODE_CIRCULAR
- LL_DMA_PERIPH_INCREMENT or
LL_DMA_PERIPH_NOINCREMENT
- LL_DMA_MEMORY_INCREMENT or
LL_DMA_MEMORY_NOINCREMENT
- LL_DMA_PDATAALIGN_BYTE or
LL_DMA_PDATAALIGN_HALFWORD or
LL_DMA_PDATAALIGN_WORD
- LL_DMA_MDATAALIGN_BYTE or
LL_DMA_MDATAALIGN_HALFWORD or
LL_DMA_MDATAALIGN_WORD
- LL_DMA_PRIORITY_LOW or
LL_DMA_PRIORITY_MEDIUM or
LL_DMA_PRIORITY_HIGH or
LL_DMA_PRIORITY_VERYHIGH

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CCR DIR LL_DMA_ConfigTransfer
- CCR MEM2MEM LL_DMA_ConfigTransfer
- CCR CIRC LL_DMA_ConfigTransfer
- CCR PINC LL_DMA_ConfigTransfer
- CCR MINC LL_DMA_ConfigTransfer
- CCR PSIZE LL_DMA_ConfigTransfer
- CCR MSIZE LL_DMA_ConfigTransfer
- CCR PL LL_DMA_ConfigTransfer

LL_DMA_SetDataTransferDirection

Function name

**STATIC_INLINE void LL_DMA_SetDataTransferDirection
(DMA_TypeDef * DMAx, uint32_t Channel, uint32_t Direction)**

Function description

Set Data transfer direction (read from peripheral or from memory).

Parameters

- **DMAx:** DMAx Instance
- **Channel:** This parameter can be one of the following values:
 - LL_DMA_CHANNEL_1
 - LL_DMA_CHANNEL_2
 - LL_DMA_CHANNEL_3
 - LL_DMA_CHANNEL_4
 - LL_DMA_CHANNEL_5
 - LL_DMA_CHANNEL_6
 - LL_DMA_CHANNEL_7
- **Direction:** This parameter can be one of the following values:
 - LL_DMA_DIRECTION_PERIPH_TO_MEMORY
 - LL_DMA_DIRECTION_MEMORY_TO_PERIPH
 - LL_DMA_DIRECTION_MEMORY_TO_MEMORY

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CCR DIR LL_DMA_SetDataTransferDirection
- CCR MEM2MEM LL_DMA_SetDataTransferDirection

LL_DMA_GetDataTransferDirection

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_GetDataTransferDirection (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Get Data transfer direction (read from peripheral or from memory).
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_DIRECTION_PERIPH_TO_MEMORY – LL_DMA_DIRECTION_MEMORY_TO_PERIPH – LL_DMA_DIRECTION_MEMORY_TO_MEMORY
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR DIR LL_DMA_GetDataTransferDirection • CCR MEM2MEM LL_DMA_GetDataTransferDirection

LL_DMA_SetMode

Function name	<code>__STATIC_INLINE void LL_DMA_SetMode (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t Mode)</code>
Function description	Set DMA mode circular or normal.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7 • Mode: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_MODE_NORMAL – LL_DMA_MODE_CIRCULAR
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • The circular buffer mode cannot be used if the memory-to-memory data transfer is configured on the selected Channel.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR CIRC LL_DMA_SetMode

LL_DMA_GetMode

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_GetMode (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Get DMA mode circular or normal.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_MODE_NORMAL – LL_DMA_MODE_CIRCULAR
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR CIRC LL_DMA_GetMode

LL_DMA_SetPeriphIncMode

Function name	<code>__STATIC_INLINE void LL_DMA_SetPeriphIncMode (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t PeriphOrM2MSrcIncMode)</code>
Function description	Set Peripheral increment mode.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7 • PeriphOrM2MSrcIncMode: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_PERIPH_INCREMENT – LL_DMA_PERIPH_NOINCREMENT
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR PINC LL_DMA_SetPeriphIncMode

LL_DMA_GetPeriphIncMode

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_GetPeriphIncMode (DMA_TypeDef * DMAx, uint32_t Channel)</code>
---------------	--

Function description	Get Peripheral increment mode.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_PERIPH_INCREMENT – LL_DMA_PERIPH_NOINCREMENT
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR PINC LL_DMA_GetPeriphIncMode

LL_DMA_SetMemoryIncMode

Function name	<code>__STATIC_INLINE void LL_DMA_SetMemoryIncMode(DMA_TypeDef * DMAx, uint32_t Channel, uint32_t MemoryOrM2MDstIncMode)</code>
Function description	Set Memory increment mode.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7 • MemoryOrM2MDstIncMode: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_MEMORY_INCREMENT – LL_DMA_MEMORY_NOINCREMENT
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR MINC LL_DMA_SetMemoryIncMode

LL_DMA_GetMemoryIncMode

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_GetMemoryIncMode(DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Get Memory increment mode.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1

	<ul style="list-style-type: none"> - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_MEMORY_INCREMENT - LL_DMA_MEMORY_NOINCREMENT
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR MINC LL_DMA_GetMemoryIncMode

LL_DMA_SetPeriphSize

Function name	<code>__STATIC_INLINE void LL_DMA_SetPeriphSize (DMA_TypeDef * DMax, uint32_t Channel, uint32_t PeriphOrM2MSrcDataSize)</code>
Function description	Set Peripheral size.
Parameters	<ul style="list-style-type: none"> • DMax: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7 • PeriphOrM2MSrcDataSize: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_PDATAALIGN_BYTE - LL_DMA_PDATAALIGN_HALFWORD - LL_DMA_PDATAALIGN_WORD
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR PSIZE LL_DMA_SetPeriphSize

LL_DMA_GetPeriphSize

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_GetPeriphSize (DMA_TypeDef * DMax, uint32_t Channel)</code>
Function description	Get Peripheral size.
Parameters	<ul style="list-style-type: none"> • DMax: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4

	<ul style="list-style-type: none"> - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_PDATAALIGN_BYTE - LL_DMA_PDATAALIGN_HALFWORD - LL_DMA_PDATAALIGN_WORD
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR PSIZE LL_DMA_GetPeriphSize

LL_DMA_SetMemorySize

Function name	<code>__STATIC_INLINE void LL_DMA_SetMemorySize(DMA_TypeDef * DMAx, uint32_t Channel, uint32_t MemoryOrM2MDstDataSize)</code>
Function description	Set Memory size.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7 • MemoryOrM2MDstDataSize: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_MDATAALIGN_BYTE - LL_DMA_MDATAALIGN_HALFWORD - LL_DMA_MDATAALIGN_WORD
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR MSIZE LL_DMA_SetMemorySize

LL_DMA_GetMemorySize

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_GetMemorySize(DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Get Memory size.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6

- LL_DMA_CHANNEL_7
- Return values
- **Returned:** value can be one of the following values:
 - LL_DMA_MDATAALIGN_BYTE
 - LL_DMA_MDATAALIGN_HALFWORD
 - LL_DMA_MDATAALIGN_WORD
- Reference Manual to
LL API cross
reference:
- CCR MSIZE LL_DMA_GetMemorySize

LL_DMA_SetChannelPriorityLevel

- Function name
- ```
__STATIC_INLINE void LL_DMA_SetChannelPriorityLevel
(DMA_TypeDef * DMAX, uint32_t Channel, uint32_t Priority)
```
- Function description
- Set Channel priority level.
- Parameters
- **DMAX:** DMAx Instance
  - **Channel:** This parameter can be one of the following values:
    - LL\_DMA\_CHANNEL\_1
    - LL\_DMA\_CHANNEL\_2
    - LL\_DMA\_CHANNEL\_3
    - LL\_DMA\_CHANNEL\_4
    - LL\_DMA\_CHANNEL\_5
    - LL\_DMA\_CHANNEL\_6
    - LL\_DMA\_CHANNEL\_7
  - **Priority:** This parameter can be one of the following values:
    - LL\_DMA\_PRIORITY\_LOW
    - LL\_DMA\_PRIORITY\_MEDIUM
    - LL\_DMA\_PRIORITY\_HIGH
    - LL\_DMA\_PRIORITY\_VERYHIGH
- Return values
- **None:**
- Reference Manual to  
LL API cross  
reference:
- CCR PL LL\_DMA\_SetChannelPriorityLevel

### **LL\_DMA\_GetChannelPriorityLevel**

- Function name
- ```
__STATIC_INLINE uint32_t LL_DMA_GetChannelPriorityLevel  
(DMA_TypeDef * DMAX, uint32_t Channel)
```
- Function description
- Get Channel priority level.
- Parameters
- **DMAX:** DMAx Instance
 - **Channel:** This parameter can be one of the following values:
 - LL_DMA_CHANNEL_1
 - LL_DMA_CHANNEL_2
 - LL_DMA_CHANNEL_3
 - LL_DMA_CHANNEL_4
 - LL_DMA_CHANNEL_5
 - LL_DMA_CHANNEL_6
 - LL_DMA_CHANNEL_7
- Return values
- **Returned:** value can be one of the following values:
 - LL_DMA_PRIORITY_LOW

- LL_DMA_PRIORITY_MEDIUM
- LL_DMA_PRIORITY_HIGH
- LL_DMA_PRIORITY_VERYHIGH

Reference Manual to
LL API cross
reference:

- CCR PL LL_DMA_GetChannelPriorityLevel

LL_DMA_SetDataLength

Function name	<code>_STATIC_INLINE void LL_DMA_SetDataLength (DMA_TypeDef * DMAX, uint32_t Channel, uint32_t NbData)</code>
Function description	Set Number of data to transfer.
Parameters	<ul style="list-style-type: none"> • DMAX: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7 • NbData: Between Min_Data = 0 and Max_Data = 0x0000FFFF • None:
Return values	
Notes	<ul style="list-style-type: none"> • This action has no effect if channel is enabled.
Reference Manual to LL API cross reference:	• CNDTR NDT LL_DMA_SetDataLength

LL_DMA_GetDataLength

Function name	<code>_STATIC_INLINE uint32_t LL_DMA_GetDataLength (DMA_TypeDef * DMAX, uint32_t Channel)</code>
Function description	Get Number of data to transfer.
Parameters	<ul style="list-style-type: none"> • DMAX: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Between: Min_Data = 0 and Max_Data = 0xFFFFFFFF
Notes	<ul style="list-style-type: none"> • Once the channel is enabled, the return value indicate the remaining bytes to be transmitted.
Reference Manual to LL API cross	• CNDTR NDT LL_DMA_GetDataLength

reference:

LL_DMA_ConfigAddresses

Function name	<code>__STATIC_INLINE void LL_DMA_ConfigAddresses (DMA_TypeDef * DMAX, uint32_t Channel, uint32_t SrcAddress, uint32_t DstAddress, uint32_t Direction)</code>
Function description	Configure the Source and Destination addresses.
Parameters	<ul style="list-style-type: none"> • DMAX: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7 • SrcAddress: Between Min_Data = 0 and Max_Data = 0xFFFFFFFF • DstAddress: Between Min_Data = 0 and Max_Data = 0xFFFFFFFF • Direction: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_DIRECTION_PERIPH_TO_MEMORY – LL_DMA_DIRECTION_MEMORY_TO_PERIPH – LL_DMA_DIRECTION_MEMORY_TO_MEMORY
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This API must not be called when the DMA channel is enabled. • Each IP using DMA provides an API to get directly the register address (LL_PPP_DMA_GetRegAddr).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CPAR PA LL_DMA_ConfigAddresses • CMAR MA LL_DMA_ConfigAddresses

LL_DMA_SetMemoryAddress

Function name	<code>__STATIC_INLINE void LL_DMA_SetMemoryAddress (DMA_TypeDef * DMAX, uint32_t Channel, uint32_t MemoryAddress)</code>
Function description	Set the Memory address.
Parameters	<ul style="list-style-type: none"> • DMAX: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7

- **MemoryAddress:** Between Min_Data = 0 and Max_Data = 0xFFFFFFFF
 - **None:**
 - Interface used for direction LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH only.
 - This API must not be called when the DMA channel is enabled.
 - CMAR MA LL_DMA_SetMemoryAddress
- Reference Manual to LL API cross reference:

LL_DMA_SetPeriphAddress

- | | |
|---|--|
| Function name | <code>__STATIC_INLINE void LL_DMA_SetPeriphAddress
(DMA_TypeDef * DMAx, uint32_t Channel, uint32_t PeriphAddress)</code> |
| Function description | Set the Peripheral address. |
| Parameters | <ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7 • PeriphAddress: Between Min_Data = 0 and Max_Data = 0xFFFFFFFF |
| Return values | <ul style="list-style-type: none"> • None: |
| Notes | <ul style="list-style-type: none"> • Interface used for direction LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH only. • This API must not be called when the DMA channel is enabled. |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> • CPAR PA LL_DMA_SetPeriphAddress |

LL_DMA_GetMemoryAddress

- | | |
|----------------------|--|
| Function name | <code>__STATIC_INLINE uint32_t LL_DMA_GetMemoryAddress
(DMA_TypeDef * DMAx, uint32_t Channel)</code> |
| Function description | Get Memory address. |
| Parameters | <ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 |

	<ul style="list-style-type: none"> - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Between: Min_Data = 0 and Max_Data = 0xFFFFFFFF
Notes	<ul style="list-style-type: none"> • Interface used for direction LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH only.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CMAR MA LL_DMA_GetMemoryAddress

LL_DMA_GetPeriphAddress

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_GetPeriphAddress (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Get Peripheral address.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Between: Min_Data = 0 and Max_Data = 0xFFFFFFFF
Notes	<ul style="list-style-type: none"> • Interface used for direction LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH only.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CPAR PA LL_DMA_GetPeriphAddress

LL_DMA_SetM2MSrcAddress

Function name	<code>__STATIC_INLINE void LL_DMA_SetM2MSrcAddress (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t MemoryAddress)</code>
Function description	Set the Memory to Memory Source address.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6

- LL_DMA_CHANNEL_7
- **MemoryAddress:** Between Min_Data = 0 and Max_Data = 0xFFFFFFFF
- **None:**
- Notes
 - Interface used for direction LL_DMA_DIRECTION_MEMORY_TO_MEMORY only.
 - This API must not be called when the DMA channel is enabled.
- Reference Manual to LL API cross reference:
 - CPAR PA LL_DMA_SetM2MSrcAddress

LL_DMA_SetM2MDstAddress

- | | |
|---|--|
| Function name | <code>__STATIC_INLINE void LL_DMA_SetM2MDstAddress(DMA_TypeDef * DMAx, uint32_t Channel, uint32_t MemoryAddress)</code> |
| Function description | Set the Memory to Memory Destination address. |
| Parameters | <ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7 • MemoryAddress: Between Min_Data = 0 and Max_Data = 0xFFFFFFFF |
| Return values | • None: |
| Notes | <ul style="list-style-type: none"> • Interface used for direction LL_DMA_DIRECTION_MEMORY_TO_MEMORY only. • This API must not be called when the DMA channel is enabled. |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> • CMAR MA LL_DMA_SetM2MDstAddress |

LL_DMA_GetM2MSrcAddress

- | | |
|----------------------|--|
| Function name | <code>__STATIC_INLINE uint32_t LL_DMA_GetM2MSrcAddress(DMA_TypeDef * DMAx, uint32_t Channel)</code> |
| Function description | Get the Memory to Memory Source address. |
| Parameters | <ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 |

	<ul style="list-style-type: none"> - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Between: Min_Data = 0 and Max_Data = 0xFFFFFFFF
Notes	<ul style="list-style-type: none"> • Interface used for direction LL_DMA_DIRECTION_MEMORY_TO_MEMORY only.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CPAR PA LL_DMA_GetM2MSrcAddress

LL_DMA_GetM2MDstAddress

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_GetM2MDstAddress (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Get the Memory to Memory Destination address.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • Between: Min_Data = 0 and Max_Data = 0xFFFFFFFF
Notes	<ul style="list-style-type: none"> • Interface used for direction LL_DMA_DIRECTION_MEMORY_TO_MEMORY only.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CMAR MA LL_DMA_GetM2MDstAddress

LL_DMA_IsActiveFlag_GI1

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI1 (DMA_TypeDef * DMAx)</code>
Function description	Get Channel 1 global interrupt flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

Reference Manual to
LL API cross
reference:
• ISR GIF1 LL_DMA_IsActiveFlag_GI1

LL_DMA_IsActiveFlag_GI2

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI2 (DMA_TypeDef * DMAx)</code>
Function description	Get Channel 2 global interrupt flag.

Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ISR GIF2 LL_DMA_IsActiveFlag_GI2

LL_DMA_IsActiveFlag_GI3

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI3(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 3 global interrupt flag.
Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ISR GIF3 LL_DMA_IsActiveFlag_GI3

LL_DMA_IsActiveFlag_GI4

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI4(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 4 global interrupt flag.
Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ISR GIF4 LL_DMA_IsActiveFlag_GI4

LL_DMA_IsActiveFlag_GI5

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI5(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 5 global interrupt flag.
Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ISR GIF5 LL_DMA_IsActiveFlag_GI5

LL_DMA_IsActiveFlag_GI6

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI6(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 6 global interrupt flag.
Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance

Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ISR GIF6 LL_DMA_IsActiveFlag_GI6

LL_DMA_IsActiveFlag_GI7

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_GI7(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 7 global interrupt flag.
Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ISR GIF7 LL_DMA_IsActiveFlag_GI7

LL_DMA_IsActiveFlag_TC1

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC1(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 1 transfer complete flag.
Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ISR TCIF1 LL_DMA_IsActiveFlag_TC1

LL_DMA_IsActiveFlag_TC2

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC2(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 2 transfer complete flag.
Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ISR TCIF2 LL_DMA_IsActiveFlag_TC2

LL_DMA_IsActiveFlag_TC3

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC3(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 3 transfer complete flag.
Parameters	<ul style="list-style-type: none"> DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- ISR TCIF3 LL_DMA_IsActiveFlag_TC3

LL_DMA_IsActiveFlag_TC4

Function name	<code>_STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC4(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 4 transfer complete flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- ISR TCIF4 LL_DMA_IsActiveFlag_TC4

LL_DMA_IsActiveFlag_TC5

Function name	<code>_STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC5(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 5 transfer complete flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- ISR TCIF5 LL_DMA_IsActiveFlag_TC5

LL_DMA_IsActiveFlag_TC6

Function name	<code>_STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC6(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 6 transfer complete flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- ISR TCIF6 LL_DMA_IsActiveFlag_TC6

LL_DMA_IsActiveFlag_TC7

Function name	<code>_STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC7(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 7 transfer complete flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
- ISR TCIF7 LL_DMA_IsActiveFlag_TC7

reference:

LL_DMA_IsActiveFlag_HT1

Function name **__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT1(DMA_TypeDef * DMAx)**

Function description Get Channel 1 half transfer flag.

Parameters • **DMAx:** DMAx Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
reference:

LL_DMA_IsActiveFlag_HT2

Function name **__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT2(DMA_TypeDef * DMAx)**

Function description Get Channel 2 half transfer flag.

Parameters • **DMAx:** DMAx Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
reference:

LL_DMA_IsActiveFlag_HT3

Function name **__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT3(DMA_TypeDef * DMAx)**

Function description Get Channel 3 half transfer flag.

Parameters • **DMAx:** DMAx Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
reference:

LL_DMA_IsActiveFlag_HT4

Function name **__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT4(DMA_TypeDef * DMAx)**

Function description Get Channel 4 half transfer flag.

Parameters • **DMAx:** DMAx Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
reference:

LL_DMA_IsActiveFlag_HT5

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT5(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 5 half transfer flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• ISR HTIF5 LL_DMA_IsActiveFlag_HT5

LL_DMA_IsActiveFlag_HT6

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT6(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 6 half transfer flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• ISR HTIF6 LL_DMA_IsActiveFlag_HT6

LL_DMA_IsActiveFlag_HT7

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT7(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 7 half transfer flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• ISR HTIF7 LL_DMA_IsActiveFlag_HT7

LL_DMA_IsActiveFlag_TE1

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE1(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 1 transfer error flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• ISR TEIF1 LL_DMA_IsActiveFlag_TE1

LL_DMA_IsActiveFlag_TE2

Function name **`_STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE2(DMA_TypeDef * DMAx)`**

Function description Get Channel 2 transfer error flag.

Parameters • **DMAx:** DMAx Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
• ISR TEIF2 LL_DMA_IsActiveFlag_TE2

LL_DMA_IsActiveFlag_TE3

Function name **`_STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE3(DMA_TypeDef * DMAx)`**

Function description Get Channel 3 transfer error flag.

Parameters • **DMAx:** DMAx Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
• ISR TEIF3 LL_DMA_IsActiveFlag_TE3

LL_DMA_IsActiveFlag_TE4

Function name **`_STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE4(DMA_TypeDef * DMAx)`**

Function description Get Channel 4 transfer error flag.

Parameters • **DMAx:** DMAx Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
• ISR TEIF4 LL_DMA_IsActiveFlag_TE4

LL_DMA_IsActiveFlag_TE5

Function name **`_STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE5(DMA_TypeDef * DMAx)`**

Function description Get Channel 5 transfer error flag.

Parameters • **DMAx:** DMAx Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
• ISR TEIF5 LL_DMA_IsActiveFlag_TE5

LL_DMA_IsActiveFlag_TE6

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE6(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 6 transfer error flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TEIF6 LL_DMA_IsActiveFlag_TE6

LL_DMA_IsActiveFlag_TE7

Function name	<code>__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE7(DMA_TypeDef * DMAx)</code>
Function description	Get Channel 7 transfer error flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TEIF7 LL_DMA_IsActiveFlag_TE7

LL_DMA_ClearFlag_GI1

Function name	<code>__STATIC_INLINE void LL_DMA_ClearFlag_GI1(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 1 global interrupt flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • IFCR CGIF1 LL_DMA_ClearFlag_GI1

LL_DMA_ClearFlag_GI2

Function name	<code>__STATIC_INLINE void LL_DMA_ClearFlag_GI2(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 2 global interrupt flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • IFCR CGIF2 LL_DMA_ClearFlag_GI2

LL_DMA_ClearFlag_GI3

Function name **_STATIC_INLINE void LL_DMA_ClearFlag_GI3
(DMA_TypeDef * DMAx)**

Function description Clear Channel 3 global interrupt flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CGIF3 LL_DMA_ClearFlag_GI3

LL_DMA_ClearFlag_GI4

Function name **_STATIC_INLINE void LL_DMA_ClearFlag_GI4
(DMA_TypeDef * DMAx)**

Function description Clear Channel 4 global interrupt flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CGIF4 LL_DMA_ClearFlag_GI4

LL_DMA_ClearFlag_GI5

Function name **_STATIC_INLINE void LL_DMA_ClearFlag_GI5
(DMA_TypeDef * DMAx)**

Function description Clear Channel 5 global interrupt flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CGIF5 LL_DMA_ClearFlag_GI5

LL_DMA_ClearFlag_GI6

Function name **_STATIC_INLINE void LL_DMA_ClearFlag_GI6
(DMA_TypeDef * DMAx)**

Function description Clear Channel 6 global interrupt flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CGIF6 LL_DMA_ClearFlag_GI6

LL_DMA_ClearFlag_GI7

Function name	<code>__STATIC_INLINE void LL_DMA_ClearFlag_GI7 (DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 7 global interrupt flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CGIF7 LL_DMA_ClearFlag_GI7

LL_DMA_ClearFlag_TC1

Function name	<code>__STATIC_INLINE void LL_DMA_ClearFlag_TC1 (DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 1 transfer complete flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CTCIF1 LL_DMA_ClearFlag_TC1

LL_DMA_ClearFlag_TC2

Function name	<code>__STATIC_INLINE void LL_DMA_ClearFlag_TC2 (DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 2 transfer complete flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CTCIF2 LL_DMA_ClearFlag_TC2

LL_DMA_ClearFlag_TC3

Function name	<code>__STATIC_INLINE void LL_DMA_ClearFlag_TC3 (DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 3 transfer complete flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CTCIF3 LL_DMA_ClearFlag_TC3

LL_DMA_ClearFlag_TC4

Function name **__STATIC_INLINE void LL_DMA_ClearFlag_TC4(DMA_TypeDef * DMAx)**

Function description Clear Channel 4 transfer complete flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CTCIF4 LL_DMA_ClearFlag_TC4

LL_DMA_ClearFlag_TC5

Function name **__STATIC_INLINE void LL_DMA_ClearFlag_TC5(DMA_TypeDef * DMAx)**

Function description Clear Channel 5 transfer complete flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CTCIF5 LL_DMA_ClearFlag_TC5

LL_DMA_ClearFlag_TC6

Function name **__STATIC_INLINE void LL_DMA_ClearFlag_TC6(DMA_TypeDef * DMAx)**

Function description Clear Channel 6 transfer complete flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CTCIF6 LL_DMA_ClearFlag_TC6

LL_DMA_ClearFlag_TC7

Function name **__STATIC_INLINE void LL_DMA_ClearFlag_TC7(DMA_TypeDef * DMAx)**

Function description Clear Channel 7 transfer complete flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CTCIF7 LL_DMA_ClearFlag_TC7

LL_DMA_ClearFlag_HT1

Function name	<code>_STATIC_INLINE void LL_DMA_ClearFlag_HT1(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 1 half transfer flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CHTIF1 LL_DMA_ClearFlag_HT1

LL_DMA_ClearFlag_HT2

Function name	<code>_STATIC_INLINE void LL_DMA_ClearFlag_HT2(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 2 half transfer flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CHTIF2 LL_DMA_ClearFlag_HT2

LL_DMA_ClearFlag_HT3

Function name	<code>_STATIC_INLINE void LL_DMA_ClearFlag_HT3(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 3 half transfer flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CHTIF3 LL_DMA_ClearFlag_HT3

LL_DMA_ClearFlag_HT4

Function name	<code>_STATIC_INLINE void LL_DMA_ClearFlag_HT4(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 4 half transfer flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CHTIF4 LL_DMA_ClearFlag_HT4

LL_DMA_ClearFlag_HT5

Function name **__STATIC_INLINE void LL_DMA_ClearFlag_HT5(DMA_TypeDef * DMAx)**

Function description Clear Channel 5 half transfer flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CHTIF5 LL_DMA_ClearFlag_HT5

LL_DMA_ClearFlag_HT6

Function name **__STATIC_INLINE void LL_DMA_ClearFlag_HT6(DMA_TypeDef * DMAx)**

Function description Clear Channel 6 half transfer flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CHTIF6 LL_DMA_ClearFlag_HT6

LL_DMA_ClearFlag_HT7

Function name **__STATIC_INLINE void LL_DMA_ClearFlag_HT7(DMA_TypeDef * DMAx)**

Function description Clear Channel 7 half transfer flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CHTIF7 LL_DMA_ClearFlag_HT7

LL_DMA_ClearFlag_TE1

Function name **__STATIC_INLINE void LL_DMA_ClearFlag_TE1(DMA_TypeDef * DMAx)**

Function description Clear Channel 1 transfer error flag.

Parameters • **DMAx:** DMAx Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
• IFCR CTEIF1 LL_DMA_ClearFlag_TE1

LL_DMA_ClearFlag_TE2

Function name	<code>_STATIC_INLINE void LL_DMA_ClearFlag_TE2(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 2 transfer error flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CTEIF2 LL_DMA_ClearFlag_TE2

LL_DMA_ClearFlag_TE3

Function name	<code>_STATIC_INLINE void LL_DMA_ClearFlag_TE3(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 3 transfer error flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CTEIF3 LL_DMA_ClearFlag_TE3

LL_DMA_ClearFlag_TE4

Function name	<code>_STATIC_INLINE void LL_DMA_ClearFlag_TE4(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 4 transfer error flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CTEIF4 LL_DMA_ClearFlag_TE4

LL_DMA_ClearFlag_TE5

Function name	<code>_STATIC_INLINE void LL_DMA_ClearFlag_TE5(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 5 transfer error flag.
Parameters	<ul style="list-style-type: none">• DMAx: DMAx Instance
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• IFCR CTEIF5 LL_DMA_ClearFlag_TE5

LL_DMA_ClearFlag_TE6

Function name	<code>__STATIC_INLINE void LL_DMA_ClearFlag_TE6(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 6 transfer error flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • IFCR CTEIF6 LL_DMA_ClearFlag_TE6

LL_DMA_ClearFlag_TE7

Function name	<code>__STATIC_INLINE void LL_DMA_ClearFlag_TE7(DMA_TypeDef * DMAx)</code>
Function description	Clear Channel 7 transfer error flag.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • IFCR CTEIF7 LL_DMA_ClearFlag_TE7

LL_DMA_EnableIT_TC

Function name	<code>__STATIC_INLINE void LL_DMA_EnableIT_TC (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Enable Transfer complete interrupt.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR TCIE LL_DMA_EnableIT_TC

LL_DMA_EnableIT_HT

Function name	<code>__STATIC_INLINE void LL_DMA_EnableIT_HT (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Enable Half transfer interrupt.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance

- **Channel:** This parameter can be one of the following values:
 - LL_DMA_CHANNEL_1
 - LL_DMA_CHANNEL_2
 - LL_DMA_CHANNEL_3
 - LL_DMA_CHANNEL_4
 - LL_DMA_CHANNEL_5
 - LL_DMA_CHANNEL_6
 - LL_DMA_CHANNEL_7

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CCR HTIE LL_DMA_EnableIT_HT

LL_DMA_EnableIT_TE

Function name **__STATIC_INLINE void LL_DMA_EnableIT_TE (DMA_TypeDef * DMAx, uint32_t Channel)**

Function description

Enable Transfer error interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Channel:** This parameter can be one of the following values:
 - LL_DMA_CHANNEL_1
 - LL_DMA_CHANNEL_2
 - LL_DMA_CHANNEL_3
 - LL_DMA_CHANNEL_4
 - LL_DMA_CHANNEL_5
 - LL_DMA_CHANNEL_6
 - LL_DMA_CHANNEL_7

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CCR TEIE LL_DMA_EnableIT_TE

LL_DMA_DisableIT_TC

Function name **__STATIC_INLINE void LL_DMA_DisableIT_TC (DMA_TypeDef * DMAx, uint32_t Channel)**

Function description

Disable Transfer complete interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Channel:** This parameter can be one of the following values:
 - LL_DMA_CHANNEL_1
 - LL_DMA_CHANNEL_2
 - LL_DMA_CHANNEL_3
 - LL_DMA_CHANNEL_4
 - LL_DMA_CHANNEL_5
 - LL_DMA_CHANNEL_6
 - LL_DMA_CHANNEL_7

Return values

- **None:**

Reference Manual to

- CCR TCIE LL_DMA_DisableIT_TC

LL API cross
reference:

LL_DMA_DisableIT_HT

Function name	<code>_STATIC_INLINE void LL_DMA_DisableIT_HT (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Disable Half transfer interrupt.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR HTIE LL_DMA_DisableIT_HT

LL_DMA_DisableIT_TE

Function name	<code>_STATIC_INLINE void LL_DMA_DisableIT_TE (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Disable Transfer error interrupt.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_DMA_CHANNEL_1 - LL_DMA_CHANNEL_2 - LL_DMA_CHANNEL_3 - LL_DMA_CHANNEL_4 - LL_DMA_CHANNEL_5 - LL_DMA_CHANNEL_6 - LL_DMA_CHANNEL_7
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR TEIE LL_DMA_DisableIT_TE

LL_DMA_IsEnabledIT_TC

Function name	<code>_STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_TC (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	Check if Transfer complete Interrupt is enabled.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values:

- LL_DMA_CHANNEL_1
- LL_DMA_CHANNEL_2
- LL_DMA_CHANNEL_3
- LL_DMA_CHANNEL_4
- LL_DMA_CHANNEL_5
- LL_DMA_CHANNEL_6
- LL_DMA_CHANNEL_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:

- CCR TCIE LL_DMA_IsEnabledIT_TC

LL_DMA_IsEnabledIT_HT

Function name **_STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_HT
(DMA_TypeDef * DMAx, uint32_t Channel)**

Function description Check if Half transfer Interrupt is enabled.

Parameters

- **DMAx:** DMAx Instance
- **Channel:** This parameter can be one of the following values:
 - LL_DMA_CHANNEL_1
 - LL_DMA_CHANNEL_2
 - LL_DMA_CHANNEL_3
 - LL_DMA_CHANNEL_4
 - LL_DMA_CHANNEL_5
 - LL_DMA_CHANNEL_6
 - LL_DMA_CHANNEL_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:

- CCR HTIE LL_DMA_IsEnabledIT_HT

LL_DMA_IsEnabledIT_TE

Function name **_STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_TE
(DMA_TypeDef * DMAx, uint32_t Channel)**

Function description Check if Transfer error Interrupt is enabled.

Parameters

- **DMAx:** DMAx Instance
- **Channel:** This parameter can be one of the following values:
 - LL_DMA_CHANNEL_1
 - LL_DMA_CHANNEL_2
 - LL_DMA_CHANNEL_3
 - LL_DMA_CHANNEL_4
 - LL_DMA_CHANNEL_5
 - LL_DMA_CHANNEL_6
 - LL_DMA_CHANNEL_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to
LL API cross

- CCR TEIE LL_DMA_IsEnabledIT_TE

reference:

LL_DMA_Init

Function name	<code>uint32_t LL_DMA_Init (DMA_TypeDef * DMAx, uint32_t Channel, LL_DMA_InitTypeDef * DMA_InitStruct)</code>
Function description	Initialize the DMA registers according to the specified parameters in DMA_InitStruct.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7 • DMA_InitStruct: pointer to a LL_DMA_InitTypeDef structure.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: DMA registers are initialized – ERROR: Not applicable
Notes	<ul style="list-style-type: none"> • To convert DMAx_Channeln Instance to DMAx Instance and Channeln, use helper macros : <code>_LL_DMA_GET_INSTANCE</code> <code>_LL_DMA_GET_CHANNEL</code>

LL_DMA_DeInit

Function name	<code>uint32_t LL_DMA_DeInit (DMA_TypeDef * DMAx, uint32_t Channel)</code>
Function description	De-initialize the DMA registers to their default reset values.
Parameters	<ul style="list-style-type: none"> • DMAx: DMAx Instance • Channel: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_DMA_CHANNEL_1 – LL_DMA_CHANNEL_2 – LL_DMA_CHANNEL_3 – LL_DMA_CHANNEL_4 – LL_DMA_CHANNEL_5 – LL_DMA_CHANNEL_6 – LL_DMA_CHANNEL_7 – LL_DMA_CHANNEL_ALL
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: DMA registers are de-initialized – ERROR: DMA registers are not de-initialized

LL_DMA_StructInit

Function name	<code>void LL_DMA_StructInit (LL_DMA_InitTypeDef * DMA_InitStruct)</code>
Function description	Set each LL_DMA_InitTypeDef field to default value.

Parameters	<ul style="list-style-type: none">• DMA_InitStruct: Pointer to a LL_DMA_InitTypeDef structure.
Return values	<ul style="list-style-type: none">• None:

54.3 DMA Firmware driver defines

54.3.1 DMA

CHANNEL

LL_DMA_CHANNEL_1	DMA Channel 1
LL_DMA_CHANNEL_2	DMA Channel 2
LL_DMA_CHANNEL_3	DMA Channel 3
LL_DMA_CHANNEL_4	DMA Channel 4
LL_DMA_CHANNEL_5	DMA Channel 5
LL_DMA_CHANNEL_6	DMA Channel 6
LL_DMA_CHANNEL_7	DMA Channel 7
LL_DMA_CHANNEL_ALL	DMA Channel all (used only for function)

Clear Flags Defines

LL_DMA_IFCR_CGIF1	Channel 1 global flag
LL_DMA_IFCR_CTCIF1	Channel 1 transfer complete flag
LL_DMA_IFCR_CHTIF1	Channel 1 half transfer flag
LL_DMA_IFCR_CTEIF1	Channel 1 transfer error flag
LL_DMA_IFCR_CGIF2	Channel 2 global flag
LL_DMA_IFCR_CTCIF2	Channel 2 transfer complete flag
LL_DMA_IFCR_CHTIF2	Channel 2 half transfer flag
LL_DMA_IFCR_CTEIF2	Channel 2 transfer error flag
LL_DMA_IFCR_CGIF3	Channel 3 global flag
LL_DMA_IFCR_CTCIF3	Channel 3 transfer complete flag
LL_DMA_IFCR_CHTIF3	Channel 3 half transfer flag
LL_DMA_IFCR_CTEIF3	Channel 3 transfer error flag
LL_DMA_IFCR_CGIF4	Channel 4 global flag
LL_DMA_IFCR_CTCIF4	Channel 4 transfer complete flag
LL_DMA_IFCR_CHTIF4	Channel 4 half transfer flag
LL_DMA_IFCR_CTEIF4	Channel 4 transfer error flag
LL_DMA_IFCR_CGIF5	Channel 5 global flag
LL_DMA_IFCR_CTCIF5	Channel 5 transfer complete flag
LL_DMA_IFCR_CHTIF5	Channel 5 half transfer flag
LL_DMA_IFCR_CTEIF5	Channel 5 transfer error flag

LL_DMA_IFCR_CGIF6	Channel 6 global flag
LL_DMA_IFCR_CTCIF6	Channel 6 transfer complete flag
LL_DMA_IFCR_CHTIF6	Channel 6 half transfer flag
LL_DMA_IFCR_CTEIF6	Channel 6 transfer error flag
LL_DMA_IFCR_CGIF7	Channel 7 global flag
LL_DMA_IFCR_CTCIF7	Channel 7 transfer complete flag
LL_DMA_IFCR_CHTIF7	Channel 7 half transfer flag
LL_DMA_IFCR_CTEIF7	Channel 7 transfer error flag

Transfer Direction

LL_DMA_DIRECTION_PERIPH_TO_MEMORY	Peripheral to memory direction
LL_DMA_DIRECTION_MEMORY_TO_PERIPH	Memory to peripheral direction
LL_DMA_DIRECTION_MEMORY_TO_MEMORY	Memory to memory direction

Get Flags Defines

LL_DMA_ISR_GIF1	Channel 1 global flag
LL_DMA_ISR_TCIF1	Channel 1 transfer complete flag
LL_DMA_ISR_HTIF1	Channel 1 half transfer flag
LL_DMA_ISR_TEIF1	Channel 1 transfer error flag
LL_DMA_ISR_GIF2	Channel 2 global flag
LL_DMA_ISR_TCIF2	Channel 2 transfer complete flag
LL_DMA_ISR_HTIF2	Channel 2 half transfer flag
LL_DMA_ISR_TEIF2	Channel 2 transfer error flag
LL_DMA_ISR_GIF3	Channel 3 global flag
LL_DMA_ISR_TCIF3	Channel 3 transfer complete flag
LL_DMA_ISR_HTIF3	Channel 3 half transfer flag
LL_DMA_ISR_TEIF3	Channel 3 transfer error flag
LL_DMA_ISR_GIF4	Channel 4 global flag
LL_DMA_ISR_TCIF4	Channel 4 transfer complete flag
LL_DMA_ISR_HTIF4	Channel 4 half transfer flag
LL_DMA_ISR_TEIF4	Channel 4 transfer error flag
LL_DMA_ISR_GIF5	Channel 5 global flag
LL_DMA_ISR_TCIF5	Channel 5 transfer complete flag
LL_DMA_ISR_HTIF5	Channel 5 half transfer flag
LL_DMA_ISR_TEIF5	Channel 5 transfer error flag
LL_DMA_ISR_GIF6	Channel 6 global flag
LL_DMA_ISR_TCIF6	Channel 6 transfer complete flag
LL_DMA_ISR_HTIF6	Channel 6 half transfer flag

<code>LL_DMA_ISR_TEIF6</code>	Channel 6 transfer error flag
<code>LL_DMA_ISR_GIF7</code>	Channel 7 global flag
<code>LL_DMA_ISR_TCIF7</code>	Channel 7 transfer complete flag
<code>LL_DMA_ISR_HTIF7</code>	Channel 7 half transfer flag
<code>LL_DMA_ISR_TEIF7</code>	Channel 7 transfer error flag

IT Defines

<code>LL_DMA_CCR_TCIE</code>	Transfer complete interrupt
<code>LL_DMA_CCR_HTIE</code>	Half Transfer interrupt
<code>LL_DMA_CCR_TEIE</code>	Transfer error interrupt

Memory data alignment

<code>LL_DMA_MDATAALIGN_BYTE</code>	Memory data alignment : Byte
<code>LL_DMA_MDATAALIGN_HALFWORD</code>	Memory data alignment : HalfWord
<code>LL_DMA_MDATAALIGN_WORD</code>	Memory data alignment : Word

Memory increment mode

<code>LL_DMA_MEMORY_INCREMENT</code>	Memory increment mode Enable
<code>LL_DMA_MEMORY_NOINCREMENT</code>	Memory increment mode Disable

Transfer mode

<code>LL_DMA_MODE_NORMAL</code>	Normal Mode
<code>LL_DMA_MODE_CIRCULAR</code>	Circular Mode

Peripheral data alignment

<code>LL_DMA_PDATAALIGN_BYTE</code>	Peripheral data alignment : Byte
<code>LL_DMA_PDATAALIGN_HALFWORD</code>	Peripheral data alignment : HalfWord
<code>LL_DMA_PDATAALIGN_WORD</code>	Peripheral data alignment : Word

Peripheral increment mode

<code>LL_DMA_PERIPH_INCREMENT</code>	Peripheral increment mode Enable
<code>LL_DMA_PERIPH_NOINCREMENT</code>	Peripheral increment mode Disable

Transfer Priority level

<code>LL_DMA_PRIORITY_LOW</code>	Priority level : Low
<code>LL_DMA_PRIORITY_MEDIUM</code>	Priority level : Medium
<code>LL_DMA_PRIORITY_HIGH</code>	Priority level : High
<code>LL_DMA_PRIORITY_VERYHIGH</code>	Priority level : Very_High

Convert DMAxChannely

<code>_LL_DMA_GET_INSTANCE</code>	Description: • Convert DMAx_Channely into DMAx. Parameters: • <code>_CHANNEL_INSTANCE_</code> : DMAx_Channely
-----------------------------------	--

[__LL_DMA_GET_CHANNEL](#)**Return value:**

- DMAx

Description:

- Convert DMAx_Channely into LL_DMA_CHANNEL_y.

Parameters:

- __CHANNEL_INSTANCE__: DMAx_Channely

Return value:

- LL_DMA_CHANNEL_y

[__LL_DMA_GET_CHANNEL_INSTANCE](#)**Description:**

- Convert DMA Instance DMAx and LL_DMA_CHANNEL_y into DMAx_Channely.

Parameters:

- __DMA_INSTANCE__: DMAx
- __CHANNEL__: LL_DMA_CHANNEL_y

Return value:

- DMAx_Channely

Common Write and read registers macros[LL_DMA_WriteReg](#)**Description:**

- Write a value in DMA register.

Parameters:

- __INSTANCE__: DMA Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

[LL_DMA_ReadReg](#)**Description:**

- Read a value in DMA register.

Parameters:

- __INSTANCE__: DMA Instance
- __REG__: Register to be read

Return value:

- Register: value

55 LL EXTI Generic Driver

55.1 EXTI Firmware driver registers structures

55.1.1 LL_EXTI_InitTypeDef

Data Fields

- *uint32_t Line_0_31*
- *FunctionalState LineCommand*
- *uint8_t Mode*
- *uint8_t Trigger*

Field Documentation

- ***uint32_t LL_EXTI_InitTypeDef::Line_0_31***
Specifies the EXTI lines to be enabled or disabled for Lines in range 0 to 31. This parameter can be any combination of [EXTI_LL_EC_LINE](#)
- ***FunctionalState LL_EXTI_InitTypeDef::LineCommand***
Specifies the new state of the selected EXTI lines. This parameter can be set either to ENABLE or DISABLE
- ***uint8_t LL_EXTI_InitTypeDef::Mode***
Specifies the mode for the EXTI lines. This parameter can be a value of [EXTI_LL_EC_MODE](#).
- ***uint8_t LL_EXTI_InitTypeDef::Trigger***
Specifies the trigger signal active edge for the EXTI lines. This parameter can be a value of [EXTI_LL_EC_TRIGGER](#).

55.2 EXTI Firmware driver API description

55.2.1 Detailed description of functions

LL_EXTI_EnableIT_0_31

Function name **_STATIC_INLINE void LL_EXTI_EnableIT_0_31 (uint32_t ExtiLine)**

Function description Enable ExtiLine Interrupt request for Lines in range 0 to 31.

- Parameters
- **ExtiLine:** This parameter can be one of the following values:
 - LL_EXTI_LINE_0
 - LL_EXTI_LINE_1
 - LL_EXTI_LINE_2
 - LL_EXTI_LINE_3
 - LL_EXTI_LINE_4
 - LL_EXTI_LINE_5
 - LL_EXTI_LINE_6
 - LL_EXTI_LINE_7
 - LL_EXTI_LINE_8
 - LL_EXTI_LINE_9
 - LL_EXTI_LINE_10
 - LL_EXTI_LINE_11
 - LL_EXTI_LINE_12
 - LL_EXTI_LINE_13

- LL_EXTI_LINE_14
- LL_EXTI_LINE_15
- LL_EXTI_LINE_16
- LL_EXTI_LINE_17
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_23
- LL_EXTI_LINE_24
- LL_EXTI_LINE_25
- LL_EXTI_LINE_26
- LL_EXTI_LINE_27
- LL_EXTI_LINE_28
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31
- LL_EXTI_LINE_ALL_0_31

Return values

- **None:**

Notes

- The reset value for the direct or internal lines (see RM) is set to 1 in order to enable the interrupt by default. Bits are set automatically at Power on.
- Please check each device line mapping for EXTI Line availability

**Reference Manual to
LL API cross
reference:**

- IMR IMx LL_EXTI_EnableIT_0_31

LL_EXTI_DisableIT_0_31**Function name**

_STATIC_INLINE void LL_EXTI_DisableIT_0_31 (uint32_t ExtiLine)

Function description

Disable ExtiLine Interrupt request for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:

- LL_EXTI_LINE_0
- LL_EXTI_LINE_1
- LL_EXTI_LINE_2
- LL_EXTI_LINE_3
- LL_EXTI_LINE_4
- LL_EXTI_LINE_5
- LL_EXTI_LINE_6
- LL_EXTI_LINE_7
- LL_EXTI_LINE_8
- LL_EXTI_LINE_9
- LL_EXTI_LINE_10
- LL_EXTI_LINE_11
- LL_EXTI_LINE_12
- LL_EXTI_LINE_13
- LL_EXTI_LINE_14

- LL_EXTI_LINE_15
- LL_EXTI_LINE_16
- LL_EXTI_LINE_17
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_23
- LL_EXTI_LINE_24
- LL_EXTI_LINE_25
- LL_EXTI_LINE_26
- LL_EXTI_LINE_27
- LL_EXTI_LINE_28
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31
- LL_EXTI_LINE_ALL_0_31

Return values

- **None:**

Notes

- The reset value for the direct or internal lines (see RM) is set to 1 in order to enable the interrupt by default. Bits are set automatically at Power on.
- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- IMR IMx LL_EXTI_DisableIT_0_31

LL_EXTI_IsEnabledIT_0_31

Function name **__STATIC_INLINE uint32_t LL_EXTI_IsEnabledIT_0_31
(uint32_t ExtiLine)**

Function description Indicate if ExtiLine Interrupt request is enabled for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:
 - LL_EXTI_LINE_0
 - LL_EXTI_LINE_1
 - LL_EXTI_LINE_2
 - LL_EXTI_LINE_3
 - LL_EXTI_LINE_4
 - LL_EXTI_LINE_5
 - LL_EXTI_LINE_6
 - LL_EXTI_LINE_7
 - LL_EXTI_LINE_8
 - LL_EXTI_LINE_9
 - LL_EXTI_LINE_10
 - LL_EXTI_LINE_11
 - LL_EXTI_LINE_12
 - LL_EXTI_LINE_13
 - LL_EXTI_LINE_14
 - LL_EXTI_LINE_15

- LL_EXTI_LINE_16
- LL_EXTI_LINE_17
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_23
- LL_EXTI_LINE_24
- LL_EXTI_LINE_25
- LL_EXTI_LINE_26
- LL_EXTI_LINE_27
- LL_EXTI_LINE_28
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31
- LL_EXTI_LINE_ALL_0_31

Return values

- **State:** of bit (1 or 0).

Notes

- The reset value for the direct or internal lines (see RM) is set to 1 in order to enable the interrupt by default. Bits are set automatically at Power on.
- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- IMR IMx LL_EXTI_IsEnabledIT_0_31

LL_EXTI_EnableEvent_0_31

Function name

`__STATIC_INLINE void LL_EXTI_EnableEvent_0_31 (uint32_t ExtiLine)`

Function description

Enable ExtiLine Event request for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:

- LL_EXTI_LINE_0
- LL_EXTI_LINE_1
- LL_EXTI_LINE_2
- LL_EXTI_LINE_3
- LL_EXTI_LINE_4
- LL_EXTI_LINE_5
- LL_EXTI_LINE_6
- LL_EXTI_LINE_7
- LL_EXTI_LINE_8
- LL_EXTI_LINE_9
- LL_EXTI_LINE_10
- LL_EXTI_LINE_11
- LL_EXTI_LINE_12
- LL_EXTI_LINE_13
- LL_EXTI_LINE_14
- LL_EXTI_LINE_15
- LL_EXTI_LINE_16

- LL_EXTI_LINE_17
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_23
- LL_EXTI_LINE_24
- LL_EXTI_LINE_25
- LL_EXTI_LINE_26
- LL_EXTI_LINE_27
- LL_EXTI_LINE_28
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31
- LL_EXTI_LINE_ALL_0_31

Return values

- **None:**

Notes

- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- EMR EMx LL_EXTI_EnableEvent_0_31

LL_EXTI_DisableEvent_0_31

Function name

`__STATIC_INLINE void LL_EXTI_DisableEvent_0_31 (uint32_t ExtiLine)`

Function description

Disable ExtiLine Event request for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:

- LL_EXTI_LINE_0
- LL_EXTI_LINE_1
- LL_EXTI_LINE_2
- LL_EXTI_LINE_3
- LL_EXTI_LINE_4
- LL_EXTI_LINE_5
- LL_EXTI_LINE_6
- LL_EXTI_LINE_7
- LL_EXTI_LINE_8
- LL_EXTI_LINE_9
- LL_EXTI_LINE_10
- LL_EXTI_LINE_11
- LL_EXTI_LINE_12
- LL_EXTI_LINE_13
- LL_EXTI_LINE_14
- LL_EXTI_LINE_15
- LL_EXTI_LINE_16
- LL_EXTI_LINE_17
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20

- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_23
- LL_EXTI_LINE_24
- LL_EXTI_LINE_25
- LL_EXTI_LINE_26
- LL_EXTI_LINE_27
- LL_EXTI_LINE_28
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31
- LL_EXTI_LINE_ALL_0_31

Return values

- **None:**

Notes

- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- EMR EMx LL_EXTI_DisableEvent_0_31

LL_EXTI_IsEnabledEvent_0_31

Function name **_STATIC_INLINE uint32_t LL_EXTI_IsEnabledEvent_0_31
(uint32_t ExtiLine)**

Function description Indicate if ExtiLine Event request is enabled for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:

- LL_EXTI_LINE_0
- LL_EXTI_LINE_1
- LL_EXTI_LINE_2
- LL_EXTI_LINE_3
- LL_EXTI_LINE_4
- LL_EXTI_LINE_5
- LL_EXTI_LINE_6
- LL_EXTI_LINE_7
- LL_EXTI_LINE_8
- LL_EXTI_LINE_9
- LL_EXTI_LINE_10
- LL_EXTI_LINE_11
- LL_EXTI_LINE_12
- LL_EXTI_LINE_13
- LL_EXTI_LINE_14
- LL_EXTI_LINE_15
- LL_EXTI_LINE_16
- LL_EXTI_LINE_17
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_23

- LL_EXTI_LINE_24
- LL_EXTI_LINE_25
- LL_EXTI_LINE_26
- LL_EXTI_LINE_27
- LL_EXTI_LINE_28
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31
- LL_EXTI_LINE_ALL_0_31

Return values

- **State:** of bit (1 or 0).

Notes

- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- EMR EMx LL_EXTI_IsEnabledEvent_0_31

LL_EXTI_EnableRisingTrig_0_31

Function name

**_STATIC_INLINE void LL_EXTI_EnableRisingTrig_0_31
(uint32_t ExtiLine)**

Function description

Enable ExtiLine Rising Edge Trigger for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:

- LL_EXTI_LINE_0
- LL_EXTI_LINE_1
- LL_EXTI_LINE_2
- LL_EXTI_LINE_3
- LL_EXTI_LINE_4
- LL_EXTI_LINE_5
- LL_EXTI_LINE_6
- LL_EXTI_LINE_7
- LL_EXTI_LINE_8
- LL_EXTI_LINE_9
- LL_EXTI_LINE_10
- LL_EXTI_LINE_11
- LL_EXTI_LINE_12
- LL_EXTI_LINE_13
- LL_EXTI_LINE_14
- LL_EXTI_LINE_15
- LL_EXTI_LINE_16
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31

Return values

- **None:**

Notes	<ul style="list-style-type: none"> The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a rising edge on a configurable interrupt line occurs during a write operation in the EXTI_RTSR register, the pending bit is not set. Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition. Please check each device line mapping for EXTI Line availability
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> RTSR RTx LL_EXTI_EnableRisingTrig_0_31

LL_EXTI_DisableRisingTrig_0_31

Function name	<code>__STATIC_INLINE void LL_EXTI_DisableRisingTrig_0_31(uint32_t ExtiLine)</code>
Function description	Disable ExtiLine Rising Edge Trigger for Lines in range 0 to 31.
Parameters	<ul style="list-style-type: none"> ExtiLine: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_EXTI_LINE_0 - LL_EXTI_LINE_1 - LL_EXTI_LINE_2 - LL_EXTI_LINE_3 - LL_EXTI_LINE_4 - LL_EXTI_LINE_5 - LL_EXTI_LINE_6 - LL_EXTI_LINE_7 - LL_EXTI_LINE_8 - LL_EXTI_LINE_9 - LL_EXTI_LINE_10 - LL_EXTI_LINE_11 - LL_EXTI_LINE_12 - LL_EXTI_LINE_13 - LL_EXTI_LINE_14 - LL_EXTI_LINE_15 - LL_EXTI_LINE_16 - LL_EXTI_LINE_18 - LL_EXTI_LINE_19 - LL_EXTI_LINE_20 - LL_EXTI_LINE_21 - LL_EXTI_LINE_22 - LL_EXTI_LINE_29 - LL_EXTI_LINE_30 - LL_EXTI_LINE_31
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a rising edge on a configurable interrupt line occurs during a write operation in the EXTI_RTSR register, the pending bit is not set. Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- RTSR RTx LL_EXTI_DisableRisingTrig_0_31

LL_EXTI_IsEnabledRisingTrig_0_31

Function name	<code>__STATIC_INLINE uint32_t LL_EXTI_IsEnabledRisingTrig_0_31 (uint32_t ExtiLine)</code>
Function description	Check if rising edge trigger is enabled for Lines in range 0 to 31.
Parameters	<ul style="list-style-type: none"> • ExtiLine: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - <code>LL_EXTI_LINE_0</code> - <code>LL_EXTI_LINE_1</code> - <code>LL_EXTI_LINE_2</code> - <code>LL_EXTI_LINE_3</code> - <code>LL_EXTI_LINE_4</code> - <code>LL_EXTI_LINE_5</code> - <code>LL_EXTI_LINE_6</code> - <code>LL_EXTI_LINE_7</code> - <code>LL_EXTI_LINE_8</code> - <code>LL_EXTI_LINE_9</code> - <code>LL_EXTI_LINE_10</code> - <code>LL_EXTI_LINE_11</code> - <code>LL_EXTI_LINE_12</code> - <code>LL_EXTI_LINE_13</code> - <code>LL_EXTI_LINE_14</code> - <code>LL_EXTI_LINE_15</code> - <code>LL_EXTI_LINE_16</code> - <code>LL_EXTI_LINE_18</code> - <code>LL_EXTI_LINE_19</code> - <code>LL_EXTI_LINE_20</code> - <code>LL_EXTI_LINE_21</code> - <code>LL_EXTI_LINE_22</code> - <code>LL_EXTI_LINE_29</code> - <code>LL_EXTI_LINE_30</code> - <code>LL_EXTI_LINE_31</code>
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • Please check each device line mapping for EXTI Line availability
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RTSR RTx LL_EXTI_IsEnabledRisingTrig_0_31

LL_EXTI_EnableFallingTrig_0_31

Function name	<code>__STATIC_INLINE void LL_EXTI_EnableFallingTrig_0_31 (uint32_t ExtiLine)</code>
---------------	--

Function description	Enable ExtiLine Falling Edge Trigger for Lines in range 0 to 31.
Parameters	<ul style="list-style-type: none"> • ExtiLine: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_EXTI_LINE_0 - LL_EXTI_LINE_1 - LL_EXTI_LINE_2 - LL_EXTI_LINE_3 - LL_EXTI_LINE_4 - LL_EXTI_LINE_5 - LL_EXTI_LINE_6 - LL_EXTI_LINE_7 - LL_EXTI_LINE_8 - LL_EXTI_LINE_9 - LL_EXTI_LINE_10 - LL_EXTI_LINE_11 - LL_EXTI_LINE_12 - LL_EXTI_LINE_13 - LL_EXTI_LINE_14 - LL_EXTI_LINE_15 - LL_EXTI_LINE_16 - LL_EXTI_LINE_18 - LL_EXTI_LINE_19 - LL_EXTI_LINE_20 - LL_EXTI_LINE_21 - LL_EXTI_LINE_22 - LL_EXTI_LINE_29 - LL_EXTI_LINE_30 - LL_EXTI_LINE_31
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a falling edge on a configurable interrupt line occurs during a write operation in the EXTI_FTSR register, the pending bit is not set. Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition. • Please check each device line mapping for EXTI Line availability
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • FTSR FTx LL_EXTI_EnableFallingTrig_0_31

LL_EXTI_DisableFallingTrig_0_31

Function name `_STATIC_INLINE void LL_EXTI_DisableFallingTrig_0_31
(uint32_t ExtiLine)`

Function description Disable ExtiLine Falling Edge Trigger for Lines in range 0 to 31.

Parameters	<ul style="list-style-type: none"> • ExtiLine: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_EXTI_LINE_0 - LL_EXTI_LINE_1
------------	---

- LL_EXTI_LINE_2
- LL_EXTI_LINE_3
- LL_EXTI_LINE_4
- LL_EXTI_LINE_5
- LL_EXTI_LINE_6
- LL_EXTI_LINE_7
- LL_EXTI_LINE_8
- LL_EXTI_LINE_9
- LL_EXTI_LINE_10
- LL_EXTI_LINE_11
- LL_EXTI_LINE_12
- LL_EXTI_LINE_13
- LL_EXTI_LINE_14
- LL_EXTI_LINE_15
- LL_EXTI_LINE_16
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31

Return values

- **None:**

Notes

- The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a Falling edge on a configurable interrupt line occurs during a write operation in the EXTI_FTSR register, the pending bit is not set. Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.
- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- FTSR FTx LL_EXTI_DisableFallingTrig_0_31

LL_EXTI_IsEnabledFallingTrig_0_31

Function name

STATIC_INLINE uint32_t

LL_EXTI_IsEnabledFallingTrig_0_31 (uint32_t ExtiLine)

Function description

Check if falling edge trigger is enabled for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:

- LL_EXTI_LINE_0
- LL_EXTI_LINE_1
- LL_EXTI_LINE_2
- LL_EXTI_LINE_3
- LL_EXTI_LINE_4
- LL_EXTI_LINE_5
- LL_EXTI_LINE_6
- LL_EXTI_LINE_7

- LL_EXTI_LINE_8
- LL_EXTI_LINE_9
- LL_EXTI_LINE_10
- LL_EXTI_LINE_11
- LL_EXTI_LINE_12
- LL_EXTI_LINE_13
- LL_EXTI_LINE_14
- LL_EXTI_LINE_15
- LL_EXTI_LINE_16
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19
- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31

Return values

- **State:** of bit (1 or 0).

Notes

- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- FTSR FTx LL_EXTI_IsEnabledFallingTrig_0_31

LL_EXTI_GenerateSWI_0_31

Function name

`_STATIC_INLINE void LL_EXTI_GenerateSWI_0_31 (uint32_t ExtiLine)`

Function description

Generate a software Interrupt Event for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:

- LL_EXTI_LINE_0
- LL_EXTI_LINE_1
- LL_EXTI_LINE_2
- LL_EXTI_LINE_3
- LL_EXTI_LINE_4
- LL_EXTI_LINE_5
- LL_EXTI_LINE_6
- LL_EXTI_LINE_7
- LL_EXTI_LINE_8
- LL_EXTI_LINE_9
- LL_EXTI_LINE_10
- LL_EXTI_LINE_11
- LL_EXTI_LINE_12
- LL_EXTI_LINE_13
- LL_EXTI_LINE_14
- LL_EXTI_LINE_15
- LL_EXTI_LINE_16
- LL_EXTI_LINE_18
- LL_EXTI_LINE_19

- LL_EXTI_LINE_20
- LL_EXTI_LINE_21
- LL_EXTI_LINE_22
- LL_EXTI_LINE_29
- LL_EXTI_LINE_30
- LL_EXTI_LINE_31

Return values

- **None:**

Notes

- If the interrupt is enabled on this line in the EXTI_IMR, writing a 1 to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation. This bit is cleared by clearing the corresponding bit in the EXTI_PR register (by writing a 1 into the bit)
- Please check each device line mapping for EXTI Line availability

**Reference Manual to
LL API cross
reference:**

- SWIER SWIx LL_EXTI_GenerateSWI_0_31

LL_EXTI_IsActiveFlag_0_31

Function name `__STATIC_INLINE uint32_t LL_EXTI_IsActiveFlag_0_31(
 uint32_t ExtiLine>)`

Function description Check if the ExtLine Flag is set or not for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
 - LL_EXTI_LINE_0
 - LL_EXTI_LINE_1
 - LL_EXTI_LINE_2
 - LL_EXTI_LINE_3
 - LL_EXTI_LINE_4
 - LL_EXTI_LINE_5
 - LL_EXTI_LINE_6
 - LL_EXTI_LINE_7
 - LL_EXTI_LINE_8
 - LL_EXTI_LINE_9
 - LL_EXTI_LINE_10
 - LL_EXTI_LINE_11
 - LL_EXTI_LINE_12
 - LL_EXTI_LINE_13
 - LL_EXTI_LINE_14
 - LL_EXTI_LINE_15
 - LL_EXTI_LINE_16
 - LL_EXTI_LINE_18
 - LL_EXTI_LINE_19
 - LL_EXTI_LINE_20
 - LL_EXTI_LINE_21
 - LL_EXTI_LINE_22
 - LL_EXTI_LINE_29
 - LL_EXTI_LINE_30
 - LL_EXTI_LINE_31

Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a 1 to the bit. Please check each device line mapping for EXTI Line availability
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> PR PIFx LL_EXTI_IsActiveFlag_0_31

LL_EXTI_ReadFlag_0_31

Function name	<code>__STATIC_INLINE uint32_t LL_EXTI_ReadFlag_0_31 (uint32_t ExtiLine)</code>
Function description	Read ExtLine Combination Flag for Lines in range 0 to 31.
Parameters	<ul style="list-style-type: none"> ExtiLine: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - <code>LL_EXTI_LINE_0</code> - <code>LL_EXTI_LINE_1</code> - <code>LL_EXTI_LINE_2</code> - <code>LL_EXTI_LINE_3</code> - <code>LL_EXTI_LINE_4</code> - <code>LL_EXTI_LINE_5</code> - <code>LL_EXTI_LINE_6</code> - <code>LL_EXTI_LINE_7</code> - <code>LL_EXTI_LINE_8</code> - <code>LL_EXTI_LINE_9</code> - <code>LL_EXTI_LINE_10</code> - <code>LL_EXTI_LINE_11</code> - <code>LL_EXTI_LINE_12</code> - <code>LL_EXTI_LINE_13</code> - <code>LL_EXTI_LINE_14</code> - <code>LL_EXTI_LINE_15</code> - <code>LL_EXTI_LINE_16</code> - <code>LL_EXTI_LINE_18</code> - <code>LL_EXTI_LINE_19</code> - <code>LL_EXTI_LINE_20</code> - <code>LL_EXTI_LINE_21</code> - <code>LL_EXTI_LINE_22</code> - <code>LL_EXTI_LINE_29</code> - <code>LL_EXTI_LINE_30</code> - <code>LL_EXTI_LINE_31</code>
Return values	<ul style="list-style-type: none"> @note: This bit is set when the selected edge event arrives on the interrupt
Notes	<ul style="list-style-type: none"> This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a 1 to the bit. Please check each device line mapping for EXTI Line availability
Reference Manual to LL API cross	<ul style="list-style-type: none"> PR PIFx LL_EXTI_ReadFlag_0_31

reference:

LL_EXTI_ClearFlag_0_31

Function name **`_STATIC_INLINE void LL_EXTI_ClearFlag_0_31 (uint32_t ExtiLine)`**

Function description Clear EXTI Flags for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:

- `LL_EXTI_LINE_0`
- `LL_EXTI_LINE_1`
- `LL_EXTI_LINE_2`
- `LL_EXTI_LINE_3`
- `LL_EXTI_LINE_4`
- `LL_EXTI_LINE_5`
- `LL_EXTI_LINE_6`
- `LL_EXTI_LINE_7`
- `LL_EXTI_LINE_8`
- `LL_EXTI_LINE_9`
- `LL_EXTI_LINE_10`
- `LL_EXTI_LINE_11`
- `LL_EXTI_LINE_12`
- `LL_EXTI_LINE_13`
- `LL_EXTI_LINE_14`
- `LL_EXTI_LINE_15`
- `LL_EXTI_LINE_16`
- `LL_EXTI_LINE_18`
- `LL_EXTI_LINE_19`
- `LL_EXTI_LINE_20`
- `LL_EXTI_LINE_21`
- `LL_EXTI_LINE_22`
- `LL_EXTI_LINE_29`
- `LL_EXTI_LINE_30`
- `LL_EXTI_LINE_31`

Return values

- **None:**

Notes

- This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a 1 to the bit.
- Please check each device line mapping for EXTI Line availability

Reference Manual to
LL API cross
reference:

- PR PIFx LL_EXTI_ClearFlag_0_31

LL_EXTI_Init

Function name **`uint32_t LL_EXTI_Init (LL_EXTI_InitTypeDef * EXTI_InitStruct)`**

Function description Initialize the EXTI registers according to the specified parameters in EXTI_InitStruct.

Parameters

- **EXTI_InitStruct:** pointer to a LL_EXTI_InitTypeDef structure.

Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: EXTI registers are initialized – ERROR: not applicable
---------------	---

LL_EXTI_DeInit

Function name	uint32_t LL_EXTI_DeInit (void)
Function description	De-initialize the EXTI registers to their default reset values.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: EXTI registers are de-initialized – ERROR: not applicable

LL_EXTI_StructInit

Function name	void LL_EXTI_StructInit (LL_EXTI_InitTypeDef * EXTI_InitStruct)
Function description	Set each LL_EXTI_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • EXTI_InitStruct: Pointer to a LL_EXTI_InitTypeDef structure.
Return values	<ul style="list-style-type: none"> • None:

55.3 EXTI Firmware driver defines

55.3.1 EXTI

LINE

LL_EXTI_LINE_0	Extended line 0
LL_EXTI_LINE_1	Extended line 1
LL_EXTI_LINE_2	Extended line 2
LL_EXTI_LINE_3	Extended line 3
LL_EXTI_LINE_4	Extended line 4
LL_EXTI_LINE_5	Extended line 5
LL_EXTI_LINE_6	Extended line 6
LL_EXTI_LINE_7	Extended line 7
LL_EXTI_LINE_8	Extended line 8
LL_EXTI_LINE_9	Extended line 9
LL_EXTI_LINE_10	Extended line 10
LL_EXTI_LINE_11	Extended line 11
LL_EXTI_LINE_12	Extended line 12
LL_EXTI_LINE_13	Extended line 13
LL_EXTI_LINE_14	Extended line 14
LL_EXTI_LINE_15	Extended line 15
LL_EXTI_LINE_16	Extended line 16

LL_EXTI_LINE_17	Extended line 17
LL_EXTI_LINE_18	Extended line 18
LL_EXTI_LINE_19	Extended line 19
LL_EXTI_LINE_20	Extended line 20
LL_EXTI_LINE_21	Extended line 21
LL_EXTI_LINE_22	Extended line 22
LL_EXTI_LINE_23	Extended line 23
LL_EXTI_LINE_ALL_0_31	All Extended line not reserved
LL_EXTI_LINE_ALL	All Extended line
LL_EXTI_LINE_NONE	None Extended line

Mode

LL_EXTI_MODE_IT	Interrupt Mode
LL_EXTI_MODE_EVENT	Event Mode
LL_EXTI_MODE_IT_EVENT	Interrupt & Event Mode

Edge Trigger

LL_EXTI_TRIGGER_NONE	No Trigger Mode
LL_EXTI_TRIGGER_RISING	Trigger Rising Mode
LL_EXTI_TRIGGER_FALLING	Trigger Falling Mode
LL_EXTI_TRIGGER_RISING_FALLING	Trigger Rising & Falling Mode

Common Write and read registers Macros

LL_EXTI_WriteReg	Description:
	<ul style="list-style-type: none"> • Write a value in EXTI register.
	Parameters:
	<ul style="list-style-type: none"> • __REG__: Register to be written • __VALUE__: Value to be written in the register
	Return value:
	<ul style="list-style-type: none"> • None
LL_EXTI_ReadReg	Description:
	<ul style="list-style-type: none"> • Read a value in EXTI register.
	Parameters:
	<ul style="list-style-type: none"> • __REG__: Register to be read
	Return value:
	<ul style="list-style-type: none"> • Register: value

56 LL GPIO Generic Driver

56.1 GPIO Firmware driver registers structures

56.1.1 LL_GPIO_InitTypeDef

Data Fields

- *uint32_t Pin*
- *uint32_t Mode*
- *uint32_t Speed*
- *uint32_t OutputType*
- *uint32_t Pull*
- *uint32_t Alternate*

Field Documentation

- ***uint32_t LL_GPIO_InitTypeDef::Pin***
Specifies the GPIO pins to be configured. This parameter can be any value of ***GPIO_LL_EC_PIN***.
- ***uint32_t LL_GPIO_InitTypeDef::Mode***
Specifies the operating mode for the selected pins. This parameter can be a value of ***GPIO_LL_EC_MODE***.GPIO HW configuration can be modified afterwards using unitary function ***LL_GPIO_SetPinMode()***.
- ***uint32_t LL_GPIO_InitTypeDef::Speed***
Specifies the speed for the selected pins. This parameter can be a value of ***GPIO_LL_EC_SPEED***.GPIO HW configuration can be modified afterwards using unitary function ***LL_GPIO_SetPinSpeed()***.
- ***uint32_t LL_GPIO_InitTypeDef::OutputType***
Specifies the operating output type for the selected pins. This parameter can be a value of ***GPIO_LL_EC_OUTPUT***.GPIO HW configuration can be modified afterwards using unitary function ***LL_GPIO_SetPinOutputType()***.
- ***uint32_t LL_GPIO_InitTypeDef::Pull***
Specifies the operating Pull-up/Pull down for the selected pins. This parameter can be a value of ***GPIO_LL_EC_PULL***.GPIO HW configuration can be modified afterwards using unitary function ***LL_GPIO_SetPinPull()***.
- ***uint32_t LL_GPIO_InitTypeDef::Alternate***
Specifies the Peripheral to be connected to the selected pins. This parameter can be a value of ***GPIO_LL_EC_AF***.GPIO HW configuration can be modified afterwards using unitary function ***LL_GPIO_SetAFPin_0_7()*** and ***LL_GPIO_SetAFPin_8_15()***.

56.2 GPIO Firmware driver API description

56.2.1 Detailed description of functions

LL_GPIO_SetPinMode

Function name **_STATIC_INLINE void LL_GPIO_SetPinMode (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Mode)**

Function description Configure gpio mode for a dedicated pin on dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:

- LL_GPIO_PIN_0
- LL_GPIO_PIN_1
- LL_GPIO_PIN_2
- LL_GPIO_PIN_3
- LL_GPIO_PIN_4
- LL_GPIO_PIN_5
- LL_GPIO_PIN_6
- LL_GPIO_PIN_7
- LL_GPIO_PIN_8
- LL_GPIO_PIN_9
- LL_GPIO_PIN_10
- LL_GPIO_PIN_11
- LL_GPIO_PIN_12
- LL_GPIO_PIN_13
- LL_GPIO_PIN_14
- LL_GPIO_PIN_15
- **Mode:** This parameter can be one of the following values:
 - LL_GPIO_MODE_INPUT
 - LL_GPIO_MODE_OUTPUT
 - LL_GPIO_MODE_ALTERNATE
 - LL_GPIO_MODE_ANALOG
- **None:**
- Notes
 - I/O mode can be Input mode, General purpose output, Alternate function mode or Analog.
 - Warning: only one pin can be passed as parameter.
- Reference Manual to LL API cross reference:
 - MODER MODEy LL_GPIO_SetPinMode

LL_GPIO_SetPinMode

Function name	<code>_STATIC_INLINE uint32_t LL_GPIO_SetPinMode (GPIO_TypeDef * GPIOx, uint32_t Pin)</code>
Function description	Return gpio mode for a dedicated pin on dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6 - LL_GPIO_PIN_7 - LL_GPIO_PIN_8 - LL_GPIO_PIN_9 - LL_GPIO_PIN_10 - LL_GPIO_PIN_11 - LL_GPIO_PIN_12 - LL_GPIO_PIN_13

	<ul style="list-style-type: none"> - LL_GPIO_PIN_14 - LL_GPIO_PIN_15
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_MODE_INPUT - LL_GPIO_MODE_OUTPUT - LL_GPIO_MODE_ALTERNATE - LL_GPIO_MODE_ANALOG
Notes	<ul style="list-style-type: none"> • I/O mode can be Input mode, General purpose output, Alternate function mode or Analog. • Warning: only one pin can be passed as parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • MODER MODEy LL_GPIO_GetPinMode

LL_GPIO_SetPinOutputType

Function name	<code>__STATIC_INLINE void LL_GPIO_SetPinOutputType(GPIO_TypeDef * GPIOx, uint32_t PinMask, uint32_t OutputType)</code>
Function description	Configure gpio output type for several pins on dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • PinMask: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6 - LL_GPIO_PIN_7 - LL_GPIO_PIN_8 - LL_GPIO_PIN_9 - LL_GPIO_PIN_10 - LL_GPIO_PIN_11 - LL_GPIO_PIN_12 - LL_GPIO_PIN_13 - LL_GPIO_PIN_14 - LL_GPIO_PIN_15 - LL_GPIO_PIN_ALL • OutputType: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_OUTPUT_PUSHPULL - LL_GPIO_OUTPUT_OPENDRAIN
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Output type as to be set when gpio pin is in output or alternate modes. Possible type are Push-pull or Open-drain.
Reference Manual to LL API cross	<ul style="list-style-type: none"> • OTYPER OTy LL_GPIO_SetPinOutputType

reference:

LL_GPIO_SetPinOutputType

Function name	<code>__STATIC_INLINE uint32_t LL_GPIO_SetPinOutputType (GPIO_TypeDef * GPIOx, uint32_t Pin)</code>
Function description	Return gpio output type for several pins on dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6 - LL_GPIO_PIN_7 - LL_GPIO_PIN_8 - LL_GPIO_PIN_9 - LL_GPIO_PIN_10 - LL_GPIO_PIN_11 - LL_GPIO_PIN_12 - LL_GPIO_PIN_13 - LL_GPIO_PIN_14 - LL_GPIO_PIN_15 - LL_GPIO_PIN_ALL
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_OUTPUT_PUSHPULL - LL_GPIO_OUTPUT_OPENDRAIN
Notes	<ul style="list-style-type: none"> • Output type as to be set when gpio pin is in output or alternate modes. Possible type are Push-pull or Open-drain. • Warning: only one pin can be passed as parameter.
Reference Manual to LL API cross reference:	OTYPER OTy LL_GPIO_SetPinOutputType

LL_GPIO_SetPinSpeed

Function name	<code>__STATIC_INLINE void LL_GPIO_SetPinSpeed (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Speed)</code>
Function description	Configure gpio speed for a dedicated pin on dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6

- LL_GPIO_PIN_7
- LL_GPIO_PIN_8
- LL_GPIO_PIN_9
- LL_GPIO_PIN_10
- LL_GPIO_PIN_11
- LL_GPIO_PIN_12
- LL_GPIO_PIN_13
- LL_GPIO_PIN_14
- LL_GPIO_PIN_15
- **Speed:** This parameter can be one of the following values:
 - LL_GPIO_SPEED_FREQ_LOW
 - LL_GPIO_SPEED_FREQ_MEDIUM
 - LL_GPIO_SPEED_FREQ_HIGH
 - LL_GPIO_SPEED_FREQ VERY_HIGH
- **None:**
- Notes
 - I/O speed can be Low, Medium, Fast or High speed.
 - Warning: only one pin can be passed as parameter.
 - Refer to datasheet for frequency specifications and the power supply and load conditions for each speed.
- Reference Manual to LL API cross reference:
 - OSPEEDR OSPEEDy LL_GPIO_SetPinSpeed

LL_GPIO_SetPinSpeed

Function name	<code>__STATIC_INLINE uint32_t LL_GPIO_SetPinSpeed (GPIO_TypeDef * GPIOx, uint32_t Pin)</code>
Function description	Return gpio speed for a dedicated pin on dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6 - LL_GPIO_PIN_7 - LL_GPIO_PIN_8 - LL_GPIO_PIN_9 - LL_GPIO_PIN_10 - LL_GPIO_PIN_11 - LL_GPIO_PIN_12 - LL_GPIO_PIN_13 - LL_GPIO_PIN_14 - LL_GPIO_PIN_15
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_SPEED_FREQ_LOW - LL_GPIO_SPEED_FREQ_MEDIUM - LL_GPIO_SPEED_FREQ_HIGH

- Notes
- LL_GPIO_SPEED_FREQ_VERY_HIGH
 - I/O speed can be Low, Medium, Fast or High speed.
 - Warning: only one pin can be passed as parameter.
 - Refer to datasheet for frequency specifications and the power supply and load conditions for each speed.
- Reference Manual to
LL API cross
reference:
- OSPEEDR OSPEEDY LL_GPIO_GetPinSpeed

LL_GPIO_SetPinPull

Function name	<code>__STATIC_INLINE void LL_GPIO_SetPinPull (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Pull)</code>
Function description	Configure gpio pull-up or pull-down for a dedicated pin on a dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6 - LL_GPIO_PIN_7 - LL_GPIO_PIN_8 - LL_GPIO_PIN_9 - LL_GPIO_PIN_10 - LL_GPIO_PIN_11 - LL_GPIO_PIN_12 - LL_GPIO_PIN_13 - LL_GPIO_PIN_14 - LL_GPIO_PIN_15 • Pull: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PULL_NO - LL_GPIO_PULL_UP - LL_GPIO_PULL_DOWN
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Warning: only one pin can be passed as parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • PUPDR PUPDy LL_GPIO_SetPinPull

LL_GPIO_SetPinPull

Function name	<code>__STATIC_INLINE uint32_t LL_GPIO_SetPinPull (GPIO_TypeDef * GPIOx, uint32_t Pin)</code>
Function description	Return gpio pull-up or pull-down for a dedicated pin on a dedicated port.

Parameters	<ul style="list-style-type: none"> GPIOx: GPIO Port Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6 - LL_GPIO_PIN_7 - LL_GPIO_PIN_8 - LL_GPIO_PIN_9 - LL_GPIO_PIN_10 - LL_GPIO_PIN_11 - LL_GPIO_PIN_12 - LL_GPIO_PIN_13 - LL_GPIO_PIN_14 - LL_GPIO_PIN_15
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PULL_NO - LL_GPIO_PULL_UP - LL_GPIO_PULL_DOWN
Notes	<ul style="list-style-type: none"> Warning: only one pin can be passed as parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> PUPDR PUPDy LL_GPIO_SetPinPull

LL_GPIO_SetAFPin_0_7

Function name	<code>__STATIC_INLINE void LL_GPIO_SetAFPin_0_7 (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Alternate)</code>
Function description	Configure gpio alternate function of a dedicated pin from 0 to 7 for a dedicated port.
Parameters	<ul style="list-style-type: none"> GPIOx: GPIO Port Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6 - LL_GPIO_PIN_7 Alternate: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_AF_0 - LL_GPIO_AF_1 - LL_GPIO_AF_2 - LL_GPIO_AF_3 - LL_GPIO_AF_4

- LL_GPIO_AF_5
- LL_GPIO_AF_6
- LL_GPIO_AF_7
- LL_GPIO_AF_8
- LL_GPIO_AF_9
- LL_GPIO_AF_10
- LL_GPIO_AF_11
- LL_GPIO_AF_12
- LL_GPIO_AF_13
- LL_GPIO_AF_14
- LL_GPIO_AF_15

Return values

- **None:**

Notes

- Possible values are from AF0 to AF15 depending on target.
- Warning: only one pin can be passed as parameter.

Reference Manual to
LL API cross
reference:

- AFRL AFSELy LL_GPIO_SetAFPin_0_7

LL_GPIO_GetAFPin_0_7

Function name

```
STATIC_INLINE uint32_t LL_GPIO_GetAFPin_0_7  
(GPIO_TypeDef * GPIOx, uint32_t Pin)
```

Function description

Return gpio alternate function of a dedicated pin from 0 to 7 for a dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
 - LL_GPIO_PIN_0
 - LL_GPIO_PIN_1
 - LL_GPIO_PIN_2
 - LL_GPIO_PIN_3
 - LL_GPIO_PIN_4
 - LL_GPIO_PIN_5
 - LL_GPIO_PIN_6
 - LL_GPIO_PIN_7

Return values

- **Returned:** value can be one of the following values:
 - LL_GPIO_AF_0
 - LL_GPIO_AF_1
 - LL_GPIO_AF_2
 - LL_GPIO_AF_3
 - LL_GPIO_AF_4
 - LL_GPIO_AF_5
 - LL_GPIO_AF_6
 - LL_GPIO_AF_7
 - LL_GPIO_AF_8
 - LL_GPIO_AF_9
 - LL_GPIO_AF_10
 - LL_GPIO_AF_11
 - LL_GPIO_AF_12
 - LL_GPIO_AF_13
 - LL_GPIO_AF_14

- LL_GPIO_AF_15
 - AFRL AFSELy LL_GPIO_SetAFPin_0_7
- Reference Manual to
LL API cross
reference:

LL_GPIO_SetAFPin_8_15

Function name	<code>_STATIC_INLINE void LL_GPIO_SetAFPin_8_15 (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Alternate)</code>
Function description	Configure gpio alternate function of a dedicated pin from 8 to 15 for a dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_8 - LL_GPIO_PIN_9 - LL_GPIO_PIN_10 - LL_GPIO_PIN_11 - LL_GPIO_PIN_12 - LL_GPIO_PIN_13 - LL_GPIO_PIN_14 - LL_GPIO_PIN_15 • Alternate: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_AF_0 - LL_GPIO_AF_1 - LL_GPIO_AF_2 - LL_GPIO_AF_3 - LL_GPIO_AF_4 - LL_GPIO_AF_5 - LL_GPIO_AF_6 - LL_GPIO_AF_7 - LL_GPIO_AF_8 - LL_GPIO_AF_9 - LL_GPIO_AF_10 - LL_GPIO_AF_11 - LL_GPIO_AF_12 - LL_GPIO_AF_13 - LL_GPIO_AF_14 - LL_GPIO_AF_15
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Possible values are from AF0 to AF15 depending on target. • Warning: only one pin can be passed as parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • AFRH AFSELy LL_GPIO_SetAFPin_8_15

LL_GPIO_GetAFPin_8_15

Function name	<code>_STATIC_INLINE uint32_t LL_GPIO_GetAFPin_8_15 (GPIO_TypeDef * GPIOx, uint32_t Pin)</code>
---------------	---

Function description	Return gpio alternate function of a dedicated pin from 8 to 15 for a dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • Pin: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_8 - LL_GPIO_PIN_9 - LL_GPIO_PIN_10 - LL_GPIO_PIN_11 - LL_GPIO_PIN_12 - LL_GPIO_PIN_13 - LL_GPIO_PIN_14 - LL_GPIO_PIN_15
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_GPIO_AF_0 - LL_GPIO_AF_1 - LL_GPIO_AF_2 - LL_GPIO_AF_3 - LL_GPIO_AF_4 - LL_GPIO_AF_5 - LL_GPIO_AF_6 - LL_GPIO_AF_7 - LL_GPIO_AF_8 - LL_GPIO_AF_9 - LL_GPIO_AF_10 - LL_GPIO_AF_11 - LL_GPIO_AF_12 - LL_GPIO_AF_13 - LL_GPIO_AF_14 - LL_GPIO_AF_15
Notes	<ul style="list-style-type: none"> • Possible values are from AF0 to AF15 depending on target.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • AFRH AFSELy LL_GPIO_GetAFPin_8_15

LL_GPIO_LockPin

Function name	<code>__STATIC_INLINE void LL_GPIO_LockPin (GPIO_TypeDef * GPIOx, uint32_t PinMask)</code>
Function description	Lock configuration of several pins for a dedicated port.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • PinMask: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_GPIO_PIN_0 - LL_GPIO_PIN_1 - LL_GPIO_PIN_2 - LL_GPIO_PIN_3 - LL_GPIO_PIN_4 - LL_GPIO_PIN_5 - LL_GPIO_PIN_6 - LL_GPIO_PIN_7

- LL_GPIO_PIN_8
- LL_GPIO_PIN_9
- LL_GPIO_PIN_10
- LL_GPIO_PIN_11
- LL_GPIO_PIN_12
- LL_GPIO_PIN_13
- LL_GPIO_PIN_14
- LL_GPIO_PIN_15
- LL_GPIO_PIN_ALL

Return values

- **None:**

Notes

- When the lock sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next reset.
- Each lock bit freezes a specific configuration register (control and alternate function registers).

Reference Manual to
LL API cross
reference:

- LCKR LCKK LL_GPIO_LockPin

LL_GPIO_IsPinLocked

Function name

`__STATIC_INLINE uint32_t LL_GPIO_IsPinLocked
(GPIO_TypeDef * GPIOx, uint32_t PinMask)`

Function description

Return 1 if all pins passed as parameter, of a dedicated port, are locked.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
 - LL_GPIO_PIN_0
 - LL_GPIO_PIN_1
 - LL_GPIO_PIN_2
 - LL_GPIO_PIN_3
 - LL_GPIO_PIN_4
 - LL_GPIO_PIN_5
 - LL_GPIO_PIN_6
 - LL_GPIO_PIN_7
 - LL_GPIO_PIN_8
 - LL_GPIO_PIN_9
 - LL_GPIO_PIN_10
 - LL_GPIO_PIN_11
 - LL_GPIO_PIN_12
 - LL_GPIO_PIN_13
 - LL_GPIO_PIN_14
 - LL_GPIO_PIN_15
 - LL_GPIO_PIN_ALL

Return values

- **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:

- LCKR LCKY LL_GPIO_IsPinLocked

LL_GPIO_IsAnyPinLocked

Function name	<code>__STATIC_INLINE uint32_t LL_GPIO_IsAnyPinLocked (GPIO_TypeDef * GPIOx)</code>
Function description	Return 1 if one of the pin of a dedicated port is locked.
Parameters	<ul style="list-style-type: none">GPIOx: GPIO Port
Return values	<ul style="list-style-type: none">State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">LCKR LCKK LL_GPIO_IsAnyPinLocked

LL_GPIO_ReadInputPort

Function name	<code>__STATIC_INLINE uint32_t LL_GPIO_ReadInputPort (GPIO_TypeDef * GPIOx)</code>
Function description	Return full input data register value for a dedicated port.
Parameters	<ul style="list-style-type: none">GPIOx: GPIO Port
Return values	<ul style="list-style-type: none">Input: data register value of port
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">IDR IDy LL_GPIO_ReadInputPort

LL_GPIO_IsInputPinSet

Function name	<code>__STATIC_INLINE uint32_t LL_GPIO_IsInputPinSet (GPIO_TypeDef * GPIOx, uint32_t PinMask)</code>
Function description	Return if input data level for several pins of dedicated port is high or low.
Parameters	<ul style="list-style-type: none">GPIOx: GPIO PortPinMask: This parameter can be a combination of the following values:<ul style="list-style-type: none">- LL_GPIO_PIN_0- LL_GPIO_PIN_1- LL_GPIO_PIN_2- LL_GPIO_PIN_3- LL_GPIO_PIN_4- LL_GPIO_PIN_5- LL_GPIO_PIN_6- LL_GPIO_PIN_7- LL_GPIO_PIN_8- LL_GPIO_PIN_9- LL_GPIO_PIN_10- LL_GPIO_PIN_11- LL_GPIO_PIN_12- LL_GPIO_PIN_13- LL_GPIO_PIN_14- LL_GPIO_PIN_15- LL_GPIO_PIN_ALL

Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> IDR IDy LL_GPIO_IsInputPinSet

LL_GPIO_WriteOutputPort

Function name	<code>__STATIC_INLINE void LL_GPIO_WriteOutputPort(GPIO_TypeDef * GPIOx, uint32_t PortValue)</code>
Function description	Write output data register for the port.
Parameters	<ul style="list-style-type: none"> GPIOx: GPIO Port PortValue: Level value for each pin of the port
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ODR ODy LL_GPIO_WriteOutputPort

LL_GPIO_ReadOutputPort

Function name	<code>__STATIC_INLINE uint32_t LL_GPIO_ReadOutputPort(GPIO_TypeDef * GPIOx)</code>
Function description	Return full output data register value for a dedicated port.
Parameters	<ul style="list-style-type: none"> GPIOx: GPIO Port
Return values	<ul style="list-style-type: none"> Output: data register value of port
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ODR ODy LL_GPIO_ReadOutputPort

LL_GPIO_IsOutputPinSet

Function name	<code>__STATIC_INLINE uint32_t LL_GPIO_IsOutputPinSet(GPIO_TypeDef * GPIOx, uint32_t PinMask)</code>
Function description	Return if input data level for several pins of dedicated port is high or low.
Parameters	<ul style="list-style-type: none"> GPIOx: GPIO Port PinMask: This parameter can be a combination of the following values: <ul style="list-style-type: none"> – LL_GPIO_PIN_0 – LL_GPIO_PIN_1 – LL_GPIO_PIN_2 – LL_GPIO_PIN_3 – LL_GPIO_PIN_4 – LL_GPIO_PIN_5 – LL_GPIO_PIN_6 – LL_GPIO_PIN_7 – LL_GPIO_PIN_8 – LL_GPIO_PIN_9 – LL_GPIO_PIN_10

- LL_GPIO_PIN_11
- LL_GPIO_PIN_12
- LL_GPIO_PIN_13
- LL_GPIO_PIN_14
- LL_GPIO_PIN_15
- LL_GPIO_PIN_ALL

Return values

- **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:

- ODR ODR LL_GPIO_IsOutputPinSet

LL_GPIO_SetOutputPin

Function name

```
STATIC_INLINE void LL_GPIO_SetOutputPin  
(GPIO_TypeDef * GPIOx, uint32_t PinMask)
```

Function description

Set several pins to high level on dedicated gpio port.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
 - LL_GPIO_PIN_0
 - LL_GPIO_PIN_1
 - LL_GPIO_PIN_2
 - LL_GPIO_PIN_3
 - LL_GPIO_PIN_4
 - LL_GPIO_PIN_5
 - LL_GPIO_PIN_6
 - LL_GPIO_PIN_7
 - LL_GPIO_PIN_8
 - LL_GPIO_PIN_9
 - LL_GPIO_PIN_10
 - LL_GPIO_PIN_11
 - LL_GPIO_PIN_12
 - LL_GPIO_PIN_13
 - LL_GPIO_PIN_14
 - LL_GPIO_PIN_15
 - LL_GPIO_PIN_ALL

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- BSRR BSy LL_GPIO_SetOutputPin

LL_GPIO_ResetOutputPin

Function name

```
STATIC_INLINE void LL_GPIO_ResetOutputPin  
(GPIO_TypeDef * GPIOx, uint32_t PinMask)
```

Function description

Set several pins to low level on dedicated gpio port.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:

- LL_GPIO_PIN_0
- LL_GPIO_PIN_1
- LL_GPIO_PIN_2
- LL_GPIO_PIN_3
- LL_GPIO_PIN_4
- LL_GPIO_PIN_5
- LL_GPIO_PIN_6
- LL_GPIO_PIN_7
- LL_GPIO_PIN_8
- LL_GPIO_PIN_9
- LL_GPIO_PIN_10
- LL_GPIO_PIN_11
- LL_GPIO_PIN_12
- LL_GPIO_PIN_13
- LL_GPIO_PIN_14
- LL_GPIO_PIN_15
- LL_GPIO_PIN_ALL

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- BRR BRy LL_GPIO_ResetOutputPin
- BSRR BRy LL_GPIO_ResetOutputPin

LL_GPIO_TogglePin

Function name **`_STATIC_INLINE void LL_GPIO_TogglePin (GPIO_TypeDef *
GPIOx, uint32_t PinMask)`**

Function description Toggle data value for several pin of dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:

- LL_GPIO_PIN_0
- LL_GPIO_PIN_1
- LL_GPIO_PIN_2
- LL_GPIO_PIN_3
- LL_GPIO_PIN_4
- LL_GPIO_PIN_5
- LL_GPIO_PIN_6
- LL_GPIO_PIN_7
- LL_GPIO_PIN_8
- LL_GPIO_PIN_9
- LL_GPIO_PIN_10
- LL_GPIO_PIN_11
- LL_GPIO_PIN_12
- LL_GPIO_PIN_13
- LL_GPIO_PIN_14
- LL_GPIO_PIN_15
- LL_GPIO_PIN_ALL

Return values

- **None:**

Reference Manual to
LL API cross

- ODR ODy LL_GPIO_TogglePin

reference:

LL_GPIO_DeInit

Function name	ErrorStatus LL_GPIO_DeInit (GPIO_TypeDef * GPIOx)
Function description	De-initialize GPIO registers (Registers restored to their default values).
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: GPIO registers are de-initialized – ERROR: Wrong GPIO Port

LL_GPIO_Init

Function name	ErrorStatus LL_GPIO_Init (GPIO_TypeDef * GPIOx, LL_GPIO_InitTypeDef * GPIO_InitStruct)
Function description	Initialize GPIO registers according to the specified parameters in GPIO_InitStruct.
Parameters	<ul style="list-style-type: none"> • GPIOx: GPIO Port • GPIO_InitStruct: pointer to a LL_GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: GPIO registers are initialized according to GPIO_InitStruct content – ERROR: Not applicable

LL_GPIO_StructInit

Function name	void LL_GPIO_StructInit (LL_GPIO_InitTypeDef * GPIO_InitStruct)
Function description	Set each LL_GPIO_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • GPIO_InitStruct: pointer to a LL_GPIO_InitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> • None:

56.3 GPIO Firmware driver defines

56.3.1 GPIO

Alternate Function

LL_GPIO_AF_0	Select alternate function 0
LL_GPIO_AF_1	Select alternate function 1
LL_GPIO_AF_2	Select alternate function 2
LL_GPIO_AF_3	Select alternate function 3
LL_GPIO_AF_4	Select alternate function 4

LL_GPIO_AF_5	Select alternate function 5
LL_GPIO_AF_6	Select alternate function 6
LL_GPIO_AF_7	Select alternate function 7
LL_GPIO_AF_8	Select alternate function 8
LL_GPIO_AF_9	Select alternate function 9
LL_GPIO_AF_10	Select alternate function 10
LL_GPIO_AF_11	Select alternate function 11
LL_GPIO_AF_12	Select alternate function 12
LL_GPIO_AF_13	Select alternate function 13
LL_GPIO_AF_14	Select alternate function 14
LL_GPIO_AF_15	Select alternate function 15

Mode

LL_GPIO_MODE_INPUT	Select input mode
LL_GPIO_MODE_OUTPUT	Select output mode
LL_GPIO_MODE_ALTERNATE	Select alternate function mode
LL_GPIO_MODE_ANALOG	Select analog mode

Output Type

LL_GPIO_OUTPUT_PUSHPULL	Select push-pull as output type
LL_GPIO_OUTPUT_OPENDRAIN	Select open-drain as output type

PIN

LL_GPIO_PIN_0	Select pin 0
LL_GPIO_PIN_1	Select pin 1
LL_GPIO_PIN_2	Select pin 2
LL_GPIO_PIN_3	Select pin 3
LL_GPIO_PIN_4	Select pin 4
LL_GPIO_PIN_5	Select pin 5
LL_GPIO_PIN_6	Select pin 6
LL_GPIO_PIN_7	Select pin 7
LL_GPIO_PIN_8	Select pin 8
LL_GPIO_PIN_9	Select pin 9
LL_GPIO_PIN_10	Select pin 10
LL_GPIO_PIN_11	Select pin 11
LL_GPIO_PIN_12	Select pin 12
LL_GPIO_PIN_13	Select pin 13
LL_GPIO_PIN_14	Select pin 14
LL_GPIO_PIN_15	Select pin 15

`LL_GPIO_PIN_ALL` Select all pins

Pull Up Pull Down

`LL_GPIO_PULL_NO` Select I/O no pull

`LL_GPIO_PULL_UP` Select I/O pull up

`LL_GPIO_PULL_DOWN` Select I/O pull down

Output Speed

`LL_GPIO_SPEED_FREQ_LOW` Select I/O low output speed

`LL_GPIO_SPEED_FREQ_MEDIUM` Select I/O medium output speed

`LL_GPIO_SPEED_FREQ_HIGH` Select I/O fast output speed

`LL_GPIO_SPEED_FREQ_VERY_HIGH` Select I/O high output speed

Common Write and read registers Macros

`LL_GPIO_WriteReg` **Description:**

- Write a value in GPIO register.

Parameters:

- `__INSTANCE__`: GPIO Instance
- `__REG__`: Register to be written
- `__VALUE__`: Value to be written in the register

Return value:

- None

`LL_GPIO_ReadReg` **Description:**

- Read a value in GPIO register.

Parameters:

- `__INSTANCE__`: GPIO Instance
- `__REG__`: Register to be read

Return value:

- Register: value

57 LL I2C Generic Driver

57.1 I2C Firmware driver registers structures

57.1.1 LL_I2C_InitTypeDef

Data Fields

- *uint32_t PeripheralMode*
- *uint32_t ClockSpeed*
- *uint32_t DutyCycle*
- *uint32_t OwnAddress1*
- *uint32_t TypeAcknowledge*
- *uint32_t OwnAddrSize*

Field Documentation

- ***uint32_t LL_I2C_InitTypeDef::PeripheralMode***
Specifies the peripheral mode. This parameter can be a value of **I2C_LL_EC_PERIPHERAL_MODE**This feature can be modified afterwards using unitary function **LL_I2C_SetMode()**.
- ***uint32_t LL_I2C_InitTypeDef::ClockSpeed***
Specifies the clock frequency. This parameter must be set to a value lower than 400kHz (in Hz)This feature can be modified afterwards using unitary function **LL_I2C_SetClockPeriod()** or **LL_I2C_SetDutyCycle()** or **LL_I2C_SetClockSpeedMode()** or **LL_I2C_ConfigSpeed()**.
- ***uint32_t LL_I2C_InitTypeDef::DutyCycle***
Specifies the I2C fast mode duty cycle. This parameter can be a value of **I2C_LL_EC_DUTYCYCLE**This feature can be modified afterwards using unitary function **LL_I2C_SetDutyCycle()**.
- ***uint32_t LL_I2C_InitTypeDef::OwnAddress1***
Specifies the device own address 1. This parameter must be a value between Min_Data = 0x00 and Max_Data = 0x3FFThis feature can be modified afterwards using unitary function **LL_I2C_SetOwnAddress1()**.
- ***uint32_t LL_I2C_InitTypeDef::TypeAcknowledge***
Specifies the ACKnowledge or Non ACKnowledge condition after the address receive match code or next received byte. This parameter can be a value of **I2C_LL_EC_I2C_ACKNOWLEDGE**This feature can be modified afterwards using unitary function **LL_I2C_AcknowledgeNextData()**.
- ***uint32_t LL_I2C_InitTypeDef::OwnAddrSize***
Specifies the device own address 1 size (7-bit or 10-bit). This parameter can be a value of **I2C_LL_EC_OWNADDRESS1**This feature can be modified afterwards using unitary function **LL_I2C_SetOwnAddress1()**.

57.2 I2C Firmware driver API description

57.2.1 Detailed description of functions

LL_I2C_Enable

Function name	__STATIC_INLINE void LL_I2C_Enable (I2C_TypeDef * I2Cx)
Function description	Enable I2C peripheral (PE = 1).

Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 PE LL_I2C_Enable

LL_I2C_Disable

Function name	<code>__STATIC_INLINE void LL_I2C_Disable (I2C_TypeDef * I2Cx)</code>
Function description	Disable I2C peripheral (PE = 0).
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 PE LL_I2C_Disable

LL_I2C_IsEnabled

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabled (I2C_TypeDef * I2Cx)</code>
Function description	Check if the I2C peripheral is enabled or disabled.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 PE LL_I2C_IsEnabled

LL_I2C_EnableDMAReq_TX

Function name	<code>__STATIC_INLINE void LL_I2C_EnableDMAReq_TX (I2C_TypeDef * I2Cx)</code>
Function description	Enable DMA transmission requests.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 DMAEN LL_I2C_EnableDMAReq_TX

LL_I2C_DisableDMAReq_TX

Function name	<code>__STATIC_INLINE void LL_I2C_DisableDMAReq_TX (I2C_TypeDef * I2Cx)</code>
Function description	Disable DMA transmission requests.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:

- Reference Manual to
LL API cross
reference:
- CR2 DMAEN LL_I2C_DisableDMAReq_TX

LL_I2C_IsEnabledDMAReq_TX

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledDMAReq_TX (I2C_TypeDef * I2Cx)</code>
Function description	Check if DMA transmission requests are enabled or disabled.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 DMAEN LL_I2C_IsEnabledDMAReq_TX

LL_I2C_EnableDMAReq_RX

Function name	<code>__STATIC_INLINE void LL_I2C_EnableDMAReq_RX (I2C_TypeDef * I2Cx)</code>
Function description	Enable DMA reception requests.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 DMAEN LL_I2C_EnableDMAReq_RX

LL_I2C_DisableDMAReq_RX

Function name	<code>__STATIC_INLINE void LL_I2C_DisableDMAReq_RX (I2C_TypeDef * I2Cx)</code>
Function description	Disable DMA reception requests.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 DMAEN LL_I2C_DisableDMAReq_RX

LL_I2C_IsEnabledDMAReq_RX

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledDMAReq_RX (I2C_TypeDef * I2Cx)</code>
Function description	Check if DMA reception requests are enabled or disabled.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross	<ul style="list-style-type: none"> • CR2 DMAEN LL_I2C_IsEnabledDMAReq_RX

reference:

LL_I2C_DMA_GetRegAddr

Function name **_STATIC_INLINE uint32_t LL_I2C_DMA_GetRegAddr
(I2C_TypeDef * I2Cx)**

Function description Get the data register address used for DMA transfer.

Parameters • **I2Cx:** I2C Instance.

Return values • **Address:** of data register

Reference Manual to DR DR LL_I2C_DMA_GetRegAddr
LL API cross
reference:

LL_I2C_EnableClockStretching

Function name **_STATIC_INLINE void LL_I2C_EnableClockStretching
(I2C_TypeDef * I2Cx)**

Function description Enable Clock stretching.

Parameters • **I2Cx:** I2C Instance.

Return values • **None:**

Notes • This bit can only be programmed when the I2C is disabled
(PE = 0).

Reference Manual to CR1 NOSTRETCH LL_I2C_EnableClockStretching
LL API cross
reference:

LL_I2C_DisableClockStretching

Function name **_STATIC_INLINE void LL_I2C_DisableClockStretching
(I2C_TypeDef * I2Cx)**

Function description Disable Clock stretching.

Parameters • **I2Cx:** I2C Instance.

Return values • **None:**

Notes • This bit can only be programmed when the I2C is disabled
(PE = 0).

Reference Manual to CR1 NOSTRETCH LL_I2C_DisableClockStretching
LL API cross
reference:

LL_I2C_IsEnabledClockStretching

Function name **_STATIC_INLINE uint32_t LL_I2C_IsEnabledClockStretching
(I2C_TypeDef * I2Cx)**

Function description Check if Clock stretching is enabled or disabled.

Parameters • **I2Cx:** I2C Instance.

Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 NOSTRETCH LL_I2C_IsEnabledClockStretching

LL_I2C_EnableGeneralCall

Function name	<code>__STATIC_INLINE void LL_I2C_EnableGeneralCall (I2C_TypeDef * I2Cx)</code>
Function description	Enable General Call.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> When enabled the Address 0x00 is ACKed.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 ENGC LL_I2C_EnableGeneralCall

LL_I2C_DisableGeneralCall

Function name	<code>__STATIC_INLINE void LL_I2C_DisableGeneralCall (I2C_TypeDef * I2Cx)</code>
Function description	Disable General Call.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> When disabled the Address 0x00 is NACKed.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 ENGC LL_I2C_DisableGeneralCall

LL_I2C_IsEnabledGeneralCall

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledGeneralCall (I2C_TypeDef * I2Cx)</code>
Function description	Check if General Call is enabled or disabled.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 ENGC LL_I2C_IsEnabledGeneralCall

LL_I2C_SetOwnAddress1

Function name	<code>__STATIC_INLINE void LL_I2C_SetOwnAddress1 (I2C_TypeDef * I2Cx, uint32_t OwnAddress1, uint32_t OwnAddrSize)</code>
---------------	--

Function description	Set the Own Address1.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • OwnAddress1: This parameter must be a value between Min_Data=0 and Max_Data=0x3FF. • OwnAddrSize: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_I2C_OWNADDRESS1_7BIT - LL_I2C_OWNADDRESS1_10BIT
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • OAR1 ADD0 LL_I2C_SetOwnAddress1 • OAR1 ADD1_7 LL_I2C_SetOwnAddress1 • OAR1 ADD8_9 LL_I2C_SetOwnAddress1 • OAR1 ADDMODE LL_I2C_SetOwnAddress1

LL_I2C_SetOwnAddress2

Function name	_STATIC_INLINE void LL_I2C_SetOwnAddress2(I2C_TypeDef * I2Cx, uint32_t OwnAddress2)
Function description	Set the 7bits Own Address2.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • OwnAddress2: This parameter must be a value between Min_Data=0 and Max_Data=0x7F.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This action has no effect if own address2 is enabled.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • OAR2 ADD2 LL_I2C_SetOwnAddress2

LL_I2C_EnableOwnAddress2

Function name	_STATIC_INLINE void LL_I2C_EnableOwnAddress2(I2C_TypeDef * I2Cx)
Function description	Enable acknowledge on Own Address2 match address.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • OAR2 ENDUAL LL_I2C_EnableOwnAddress2

LL_I2C_DisableOwnAddress2

Function name	_STATIC_INLINE void LL_I2C_DisableOwnAddress2(I2C_TypeDef * I2Cx)
Function description	Disable acknowledge on Own Address2 match address.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:

- Reference Manual to
LL API cross
reference:
- OAR2 ENDUAL LL_I2C_DisableOwnAddress2

LL_I2C_IsEnabledOwnAddress2

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_IsEnabledOwnAddress2 (I2C_TypeDef * I2Cx)</code>
Function description	Check if Own Address1 acknowledge is enabled or disabled.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- OAR2 ENDUAL LL_I2C_IsEnabledOwnAddress2

LL_I2C_SetPeriphClock

Function name	<code>_STATIC_INLINE void LL_I2C_SetPeriphClock (I2C_TypeDef * I2Cx, uint32_t PeriphClock)</code>
Function description	Configure the Peripheral clock frequency.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • PeriphClock: Peripheral Clock (in Hz)
Return values	<ul style="list-style-type: none"> • None:

- Reference Manual to
LL API cross
reference:
- CR2 FREQ LL_I2C_SetPeriphClock

LL_I2C_GetPeriphClock

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_GetPeriphClock (I2C_TypeDef * I2Cx)</code>
Function description	Get the Peripheral clock frequency.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • Value: of Peripheral Clock (in Hz)

- Reference Manual to
LL API cross
reference:
- CR2 FREQ LL_I2C_GetPeriphClock

LL_I2C_SetDutyCycle

Function name	<code>_STATIC_INLINE void LL_I2C_SetDutyCycle (I2C_TypeDef * I2Cx, uint32_t DutyCycle)</code>
Function description	Configure the Duty cycle (Fast mode only).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • DutyCycle: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_I2C_DUTYCYCLE_2

– LL_I2C_DUTYCYCLE_16_9

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CCR DUTY LL_I2C_SetDutyCycle

LL_I2C_GetDutyCycle

Function name **__STATIC_INLINE uint32_t LL_I2C_GetDutyCycle(I2C_TypeDef * I2Cx)**

Function description Get the Duty cycle (Fast mode only).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Returned:** value can be one of the following values:
 - LL_I2C_DUTYCYCLE_2
 - LL_I2C_DUTYCYCLE_16_9

Reference Manual to
LL API cross
reference:

- CCR DUTY LL_I2C_GetDutyCycle

LL_I2C_SetClockSpeedMode

Function name **__STATIC_INLINE void LL_I2C_SetClockSpeedMode(I2C_TypeDef * I2Cx, uint32_t ClockSpeedMode)**

Function description Configure the I2C master clock speed mode.

Parameters

- **I2Cx:** I2C Instance.
- **ClockSpeedMode:** This parameter can be one of the following values:
 - LL_I2C_CLOCK_SPEED_STANDARD_MODE
 - LL_I2C_CLOCK_SPEED_FAST_MODE

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CCR FS LL_I2C_SetClockSpeedMode

LL_I2C_GetClockSpeedMode

Function name **__STATIC_INLINE uint32_t LL_I2C_GetClockSpeedMode(I2C_TypeDef * I2Cx)**

Function description Get the the I2C master speed mode.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Returned:** value can be one of the following values:
 - LL_I2C_CLOCK_SPEED_STANDARD_MODE
 - LL_I2C_CLOCK_SPEED_FAST_MODE

Reference Manual to
LL API cross
reference:

- CCR FS LL_I2C_GetClockSpeedMode

LL_I2C_SetRiseTime

Function name	<code>__STATIC_INLINE void LL_I2C_SetRiseTime (I2C_TypeDef * I2Cx, uint32_t RiseTime)</code>
Function description	Configure the SCL, SDA rising time.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • RiseTime: This parameter must be a value between Min_Data=0x02 and Max_Data=0x3F.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit can only be programmed when the I2C is disabled (PE = 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TRISE TRISE LL_I2C_SetRiseTime

LL_I2C_GetRiseTime

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_GetRiseTime (I2C_TypeDef * I2Cx)</code>
Function description	Get the SCL, SDA rising time.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x02 and Max_Data=0x3F
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TRISE TRISE LL_I2C_GetRiseTime

LL_I2C_SetClockPeriod

Function name	<code>__STATIC_INLINE void LL_I2C_SetClockPeriod (I2C_TypeDef * I2Cx, uint32_t ClockPeriod)</code>
Function description	Configure the SCL high and low period.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • ClockPeriod: This parameter must be a value between Min_Data=0x004 and Max_Data=0xFFFF, except in FAST DUTY mode where Min_Data=0x001.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit can only be programmed when the I2C is disabled (PE = 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CCR CCR LL_I2C_SetClockPeriod

LL_I2C_GetClockPeriod

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_GetClockPeriod (I2C_TypeDef * I2Cx)</code>
---------------	--

Function description	Get the SCL high and low period.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x004 and Max_Data=0xFFFF, except in FAST DUTY mode where Min_Data=0x001.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CCR CCR LL_I2C_GetClockPeriod

LL_I2C_ConfigSpeed

Function name	<code>__STATIC_INLINE void LL_I2C_ConfigSpeed (I2C_TypeDef * I2Cx, uint32_t PeriphClock, uint32_t ClockSpeed, uint32_t DutyCycle)</code>
Function description	Configure the SCL speed.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance. PeriphClock: Peripheral Clock (in Hz) ClockSpeed: This parameter must be a value lower than 400kHz (in Hz). DutyCycle: This parameter can be one of the following values: <ul style="list-style-type: none"> LL_I2C_DUTYCYCLE_2 LL_I2C_DUTYCYCLE_16_9
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> This bit can only be programmed when the I2C is disabled (PE = 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 FREQ LL_I2C_ConfigSpeed TRISE TRISE LL_I2C_ConfigSpeed CCR FS LL_I2C_ConfigSpeed CCR DUTY LL_I2C_ConfigSpeed CCR CCR LL_I2C_ConfigSpeed

LL_I2C_SetMode

Function name	<code>__STATIC_INLINE void LL_I2C_SetMode (I2C_TypeDef * I2Cx, uint32_t PeripheralMode)</code>
Function description	Configure peripheral mode.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance. PeripheralMode: This parameter can be one of the following values: <ul style="list-style-type: none"> LL_I2C_MODE_I2C LL_I2C_MODE_SMBUS_HOST LL_I2C_MODE_SMBUS_DEVICE LL_I2C_MODE_SMBUS_DEVICE_ARP
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 SMBUS LL_I2C_SetMode • CR1 SMBTYPE LL_I2C_SetMode • CR1 ENARP LL_I2C_SetMode
---	--

LL_I2C_GetMode

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_GetMode (I2C_TypeDef * I2Cx)</code>
Function description	Get peripheral mode.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_I2C_MODE_I2C - LL_I2C_MODE_SMBUS_HOST - LL_I2C_MODE_SMBUS_DEVICE - LL_I2C_MODE_SMBUS_DEVICE_ARP
Notes	<ul style="list-style-type: none"> • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 SMBUS LL_I2C_GetMode • CR1 SMBTYPE LL_I2C_GetMode • CR1 ENARP LL_I2C_GetMode

LL_I2C_EnableSMBusAlert

Function name	<code>__STATIC_INLINE void LL_I2C_EnableSMBusAlert (I2C_TypeDef * I2Cx)</code>
Function description	Enable SMBus alert (Host or Device mode)
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance. • SMBus Device mode: SMBus Alert pin is driven low and Alert Response Address Header acknowledge is enabled. SMBus Host mode: SMBus Alert pin management is supported.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 ALERT LL_I2C_EnableSMBusAlert

LL_I2C_DisableSMBusAlert

Function name	<code>__STATIC_INLINE void LL_I2C_DisableSMBusAlert (I2C_TypeDef * I2Cx)</code>
Function description	Disable SMBus alert (Host or Device mode)
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:

Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance. SMBus Device mode: SMBus Alert pin is not driven (can be used as a standard GPIO) and Alert Response Address Header acknowledge is disabled. SMBus Host mode:SMBus Alert pin management is not supported.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 ALERT LL_I2C_DisableSMBusAlert

LL_I2C_IsEnabledSMBusAlert

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledSMBusAlert (I2C_TypeDef * I2Cx)</code>
Function description	Check if SMBus alert (Host or Device mode) is enabled or disabled.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 ALERT LL_I2C_IsEnabledSMBusAlert

LL_I2C_EnableSMBusPEC

Function name	<code>__STATIC_INLINE void LL_I2C_EnableSMBusPEC (I2C_TypeDef * I2Cx)</code>
Function description	Enable SMBus Packet Error Calculation (PEC).
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 ENPEC LL_I2C_EnableSMBusPEC

LL_I2C_DisableSMBusPEC

Function name	<code>__STATIC_INLINE void LL_I2C_DisableSMBusPEC (I2C_TypeDef * I2Cx)</code>
Function description	Disable SMBus Packet Error Calculation (PEC).
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:

Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 ENPEC LL_I2C_DisableSMBusPEC

LL_I2C_IsEnabledSMBusPEC

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledSMBusPEC(I2C_TypeDef * I2Cx)</code>
Function description	Check if SMBus Packet Error Calculation (PEC) is enabled or disabled.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 ENPEC LL_I2C_IsEnabledSMBusPEC

LL_I2C_EnableIT_TX

Function name	<code>__STATIC_INLINE void LL_I2C_EnableIT_TX (I2C_TypeDef * I2Cx)</code>
Function description	Enable TXE interrupt.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ITEVTEN LL_I2C_EnableIT_TX CR2 ITBUFEN LL_I2C_EnableIT_TX

LL_I2C_DisableIT_TX

Function name	<code>__STATIC_INLINE void LL_I2C_DisableIT_TX (I2C_TypeDef * I2Cx)</code>
Function description	Disable TXE interrupt.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ITEVTEN LL_I2C_DisableIT_TX CR2 ITBUFEN LL_I2C_DisableIT_TX

LL_I2C_IsEnabledIT_TX

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_TX</code>
---------------	---

(I2C_TypeDef * I2Cx)

Function description	Check if the TXE Interrupt is enabled or disabled.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ITEVTEN LL_I2C_IsEnabledIT_TX • CR2 ITBUFEN LL_I2C_IsEnabledIT_TX

LL_I2C_EnableIT_RX

Function name	_STATIC_INLINE void LL_I2C_EnableIT_RX (I2C_TypeDef * I2Cx)
Function description	Enable RXNE interrupt.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ITEVTEN LL_I2C_EnableIT_RX • CR2 ITBUFEN LL_I2C_EnableIT_RX

LL_I2C_DisableIT_RX

Function name	_STATIC_INLINE void LL_I2C_DisableIT_RX (I2C_TypeDef * I2Cx)
Function description	Disable RXNE interrupt.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ITEVTEN LL_I2C_DisableIT_RX • CR2 ITBUFEN LL_I2C_DisableIT_RX

LL_I2C_IsEnabledIT_RX

Function name	_STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_RX (I2C_TypeDef * I2Cx)
Function description	Check if the RXNE Interrupt is enabled or disabled.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ITEVTEN LL_I2C_IsEnabledIT_RX • CR2 ITBUFEN LL_I2C_IsEnabledIT_RX

LL_I2C_EnableIT_EVT

Function name	_STATIC_INLINE void LL_I2C_EnableIT_EVT (I2C_TypeDef * I2Cx)
---------------	---

Function description	Enable Events interrupts.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Any of these events will generate interrupt : Start Bit (SB) Address sent, Address matched (ADDR) 10-bit header sent (ADD10) Stop detection (STOPF) Byte transfer finished (BTF) • Any of these events will generate interrupt if Buffer interrupts are enabled too(using unitary function LL_I2C_EnableIT_BUF()) : Receive buffer not empty (RXNE) Transmit buffer empty (TXE)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ITEVTEN LL_I2C_EnableIT_EVT

LL_I2C_DisableIT_EVT

Function name	<code>_STATIC_INLINE void LL_I2C_DisableIT_EVT (I2C_TypeDef * I2Cx)</code>
Function description	Disable Events interrupts.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Any of these events will generate interrupt : Start Bit (SB) Address sent, Address matched (ADDR) 10-bit header sent (ADD10) Stop detection (STOPF) Byte transfer finished (BTF) Receive buffer not empty (RXNE) Transmit buffer empty (TXE)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ITEVTEN LL_I2C_DisableIT_EVT

LL_I2C_IsEnabledIT_EVT

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_EVT (I2C_TypeDef * I2Cx)</code>
Function description	Check if Events interrupts are enabled or disabled.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ITEVTEN LL_I2C_IsEnabledIT_EVT

LL_I2C_EnableIT_BUF

Function name	<code>_STATIC_INLINE void LL_I2C_EnableIT_BUF (I2C_TypeDef * I2Cx)</code>
Function description	Enable Buffer interrupts.

Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Any of these Buffer events will generate interrupt if Events interrupts are enabled too(using unitary function LL_I2C_EnableIT_EVT()) : Receive buffer not empty (RXNE) Transmit buffer empty (TXE)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ITBUFEN LL_I2C_EnableIT_BUF

LL_I2C_DisableIT_BUF

Function name	<code>_STATIC_INLINE void LL_I2C_DisableIT_BUF (I2C_TypeDef * I2Cx)</code>
Function description	Disable Buffer interrupts.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Any of these Buffer events will generate interrupt : Receive buffer not empty (RXNE) Transmit buffer empty (TXE)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ITBUFEN LL_I2C_DisableIT_BUF

LL_I2C_IsEnabledIT_BUF

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_BUF (I2C_TypeDef * I2Cx)</code>
Function description	Check if Buffer interrupts are enabled or disabled.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ITBUFEN LL_I2C_IsEnabledIT_BUF

LL_I2C_EnableIT_ERR

Function name	<code>_STATIC_INLINE void LL_I2C_EnableIT_ERR (I2C_TypeDef * I2Cx)</code>
Function description	Enable Error interrupts.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance. Any of these errors will generate interrupt : Bus Error detection (BERR) Arbitration Loss (ARLO) Acknowledge

Failure(AF) Overrun/Underrun (OVR) SMBus Timeout detection (TIMEOUT) SMBus PEC error detection (PECERR) SMBus Alert pin event detection (SMBALERT)

Reference Manual to
LL API cross
reference:

- CR2 ITERREN LL_I2C_EnableIT_ERR

LL_I2C_DisableIT_ERR

Function name	<code>__STATIC_INLINE void LL_I2C_DisableIT_ERR (I2C_TypeDef * I2Cx)</code>
Function description	Disable Error interrupts.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance. • Any of these errors will generate interrupt : Bus Error detection (BERR) Arbitration Loss (ARLO) Acknowledge Failure(AF) Overrun/Underrun (OVR) SMBus Timeout detection (TIMEOUT) SMBus PEC error detection (PECERR) SMBus Alert pin event detection (SMBALERT)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ITERREN LL_I2C_DisableIT_ERR

LL_I2C_IsEnabledIT_ERR

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_ERR (I2C_TypeDef * I2Cx)</code>
Function description	Check if Error interrupts are enabled or disabled.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

Reference Manual to
LL API cross
reference:

LL_I2C_IsActiveFlag_TXE

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_TXE (I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of Transmit data register empty flag.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • RESET: When next data is written in Transmit data register. • SET: When Transmit data register is empty.

- Reference Manual to
LL API cross
reference:
- SR1 TXE LL_I2C_IsActiveFlag_TXE

LL_I2C_IsActiveFlag_BTF

Function name **_STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_BTF
(I2C_TypeDef * I2Cx)**

Function description Indicate the status of Byte Transfer Finished flag.

Parameters • **I2Cx:** I2C Instance.

Return values • **State:** of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- SR1 BTF LL_I2C_IsActiveFlag_BTF

LL_I2C_IsActiveFlag_RXNE

Function name **_STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_RXNE
(I2C_TypeDef * I2Cx)**

Function description Indicate the status of Receive data register not empty flag.

Parameters • **I2Cx:** I2C Instance.

Return values • **State:** of bit (1 or 0).

Notes • RESET: When Receive data register is read. SET: When the received data is copied in Receive data register.

- Reference Manual to
LL API cross
reference:
- SR1 RXNE LL_I2C_IsActiveFlag_RXNE

LL_I2C_IsActiveFlag_SB

Function name **_STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_SB
(I2C_TypeDef * I2Cx)**

Function description Indicate the status of Start Bit (master mode).

Parameters • **I2Cx:** I2C Instance.

Return values • **State:** of bit (1 or 0).

Notes • RESET: When No Start condition. SET: When Start condition is generated.

- Reference Manual to
LL API cross
reference:
- SR1 SB LL_I2C_IsActiveFlag_SB

LL_I2C_IsActiveFlag_ADDR

Function name **_STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_ADDR
(I2C_TypeDef * I2Cx)**

Function description Indicate the status of Address sent (master mode) or Address

	matched flag (slave mode).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • RESET: Clear default value. SET: When the address is fully sent (master mode) or when the received slave address matched with one of the enabled slave address (slave mode).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR1 ADDR LL_I2C_IsActiveFlag_ADDR

LL_I2C_IsActiveFlag_ADD10

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_ADD10(I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of 10-bit header sent (master mode).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • RESET: When no ADD10 event occurred. SET: When the master has sent the first address byte (header).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR1 ADD10 LL_I2C_IsActiveFlag_ADD10

LL_I2C_IsActiveFlag_AF

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_AF(I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of Acknowledge failure flag.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • RESET: No acknowledge failure. SET: When an acknowledge failure is received after a byte transmission.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR1 AF LL_I2C_IsActiveFlag_AF

LL_I2C_IsActiveFlag_STOP

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_STOP(I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of Stop detection flag (slave mode).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • RESET: Clear default value. SET: When a Stop condition is

detected.

Reference Manual to
LL API cross
reference:

- SR1 STOPF LL_I2C_IsActiveFlag_STOP

LL_I2C_IsActiveFlag_BERR

Function name **`__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_BERR(I2C_TypeDef * I2Cx)`**

Function description Indicate the status of Bus error flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When a misplaced Start or Stop condition is detected.

Reference Manual to
LL API cross
reference:

- SR1 BERR LL_I2C_IsActiveFlag_BERR

LL_I2C_IsActiveFlag_ARLO

Function name **`__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_ARLO(I2C_TypeDef * I2Cx)`**

Function description Indicate the status of Arbitration lost flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When arbitration lost.

Reference Manual to
LL API cross
reference:

- SR1 ARLO LL_I2C_IsActiveFlag_ARLO

LL_I2C_IsActiveFlag_OVR

Function name **`__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_OVR(I2C_TypeDef * I2Cx)`**

Function description Indicate the status of Overrun/Underrun flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When an overrun/underrun error occurs (Clock Stretching Disabled).

Reference Manual to
LL API cross
reference:

- SR1 OVR LL_I2C_IsActiveFlag_OVR

LL_I2C_IsActiveSMBusFlag_PECERR

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_PECERR (I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of SMBus PEC error flag in reception.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR1 PECERR LL_I2C_IsActiveSMBusFlag_PECERR

LL_I2C_IsActiveSMBusFlag_TIMEOUT

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_TIMEOUT (I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of SMBus Timeout detection flag.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR1 TIMEOUT LL_I2C_IsActiveSMBusFlag_TIMEOUT

LL_I2C_IsActiveSMBusFlag_ALERT

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_ALERT (I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of SMBus alert flag.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR1 SMBALERT LL_I2C_IsActiveSMBusFlag_ALERT

LL_I2C_IsActiveFlag_BUSY

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_BUSY (I2C_TypeDef * I2Cx)</code>
---------------	---

Function description	Indicate the status of Bus Busy flag.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> RESET: Clear default value. SET: When a Start condition is detected.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR2 BUSY LL_I2C_IsActiveFlag_BUSY

LL_I2C_IsActiveFlag_DUAL

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_DUAL(I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of Dual flag.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> RESET: Received address matched with OAR1. SET: Received address matched with OAR2.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR2 DUALF LL_I2C_IsActiveFlag_DUAL

LL_I2C_IsActiveSMBusFlag_SMBHOST

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_SMBHOST (I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of SMBus Host address reception (Slave mode).
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance. RESET: No SMBus Host address SET: SMBus Host address received. This status is cleared by hardware after a STOP condition or repeated START condition.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR2 SMBHOST LL_I2C_IsActiveSMBusFlag_SMBHOST

LL_I2C_IsActiveSMBusFlag_SMBDEFAULT

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_SMBDEFAULT (I2C_TypeDef * I2Cx)</code>
---------------	--

Function description	Indicate the status of SMBus Device default address reception (Slave mode).
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance. RESET: No SMBus Device default address SET: SMBus Device default address received. This status is cleared by hardware after a STOP condition or repeated START condition.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR2 SMBDEFAULT LL_I2C_IsActiveSMBusFlag_SMBDEFAULT

LL_I2C_IsActiveFlag_GENCALL

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_GENCALL(I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of General call address reception (Slave mode).
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> RESET: No General call address SET: General call address received. This status is cleared by hardware after a STOP condition or repeated START condition.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR2 GENCALL LL_I2C_IsActiveFlag_GENCALL

LL_I2C_IsActiveFlag_MSL

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_MSL(I2C_TypeDef * I2Cx)</code>
Function description	Indicate the status of Master/Slave flag.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> RESET: Slave Mode. SET: Master Mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR2 MSL LL_I2C_IsActiveFlag_MSL

LL_I2C_ClearFlag_ADDR

Function name	<code>_STATIC_INLINE void LL_I2C_ClearFlag_ADDR(I2C_TypeDef * I2Cx)</code>
---------------	--

Function description	Clear Address Matched flag.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Clearing this flag is done by a read access to the I2Cx_SR1 register followed by a read access to the I2Cx_SR2 register.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR1 ADDR LL_I2C_ClearFlag_ADDR

LL_I2C_ClearFlag_AF

Function name	<code>_STATIC_INLINE void LL_I2C_ClearFlag_AF (I2C_TypeDef * I2Cx)</code>
Function description	Clear Acknowledge failure flag.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR1 AF LL_I2C_ClearFlag_AF

LL_I2C_ClearFlag_STOP

Function name	<code>_STATIC_INLINE void LL_I2C_ClearFlag_STOP (I2C_TypeDef * I2Cx)</code>
Function description	Clear Stop detection flag.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Clearing this flag is done by a read access to the I2Cx_SR1 register followed by a write access to I2Cx_CR1 register.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR1 STOPF LL_I2C_ClearFlag_STOP CR1 PE LL_I2C_ClearFlag_STOP

LL_I2C_ClearFlag_BERR

Function name	<code>_STATIC_INLINE void LL_I2C_ClearFlag_BERR (I2C_TypeDef * I2Cx)</code>
Function description	Clear Bus error flag.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR1 BERR LL_I2C_ClearFlag_BERR

LL_I2C_ClearFlag_ARLO

Function name **_STATIC_INLINE void LL_I2C_ClearFlag_ARLO (I2C_TypeDef * I2Cx)**

Function description Clear Arbitration lost flag.

Parameters • **I2Cx:** I2C Instance.

Return values • **None:**

Reference Manual to
LL API cross
reference:
• SR1 ARLO LL_I2C_ClearFlag_ARLO

LL_I2C_ClearFlag_OVR

Function name **_STATIC_INLINE void LL_I2C_ClearFlag_OVR (I2C_TypeDef * I2Cx)**

Function description Clear Overrun/Underrun flag.

Parameters • **I2Cx:** I2C Instance.

Return values • **None:**

Reference Manual to
LL API cross
reference:
• SR1 OVR LL_I2C_ClearFlag_OVR

LL_I2C_ClearSMBusFlag_PECERR

Function name **_STATIC_INLINE void LL_I2C_ClearSMBusFlag_PECERR (I2C_TypeDef * I2Cx)**

Function description Clear SMBus PEC error flag.

Parameters • **I2Cx:** I2C Instance.

Return values • **None:**

Reference Manual to
LL API cross
reference:
• SR1 PECERR LL_I2C_ClearSMBusFlag_PECERR

LL_I2C_ClearSMBusFlag_TIMEOUT

Function name **_STATIC_INLINE void LL_I2C_ClearSMBusFlag_TIMEOUT (I2C_TypeDef * I2Cx)**

Function description Clear SMBus Timeout detection flag.

Parameters • **I2Cx:** I2C Instance.

Return values • **None:**

Notes • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to
LL API cross
reference:
• SR1 TIMEOUT LL_I2C_ClearSMBusFlag_TIMEOUT

reference:

LL_I2C_ClearSMBusFlag_ALERT

Function name	<code>_STATIC_INLINE void LL_I2C_ClearSMBusFlag_ALERT(I2C_TypeDef * I2Cx)</code>
Function description	Clear SMBus Alert flag.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR1 SMBALERT LL_I2C_ClearSMBusFlag_ALERT

LL_I2C_EnableReset

Function name	<code>_STATIC_INLINE void LL_I2C_EnableReset (I2C_TypeDef * I2Cx)</code>
Function description	Enable Reset of I2C peripheral.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 SWRST LL_I2C_EnableReset

LL_I2C_DisableReset

Function name	<code>_STATIC_INLINE void LL_I2C_DisableReset (I2C_TypeDef * I2Cx)</code>
Function description	Disable Reset of I2C peripheral.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 SWRST LL_I2C_DisableReset

LL_I2C_IsResetEnabled

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_IsResetEnabled (I2C_TypeDef * I2Cx)</code>
Function description	Check if the I2C peripheral is under reset state or not.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to LL API cross reference:
- CR1 SWRST LL_I2C_IsResetEnabled

LL_I2C_AcknowledgeNextData

Function name	<code>__STATIC_INLINE void LL_I2C_AcknowledgeNextData(I2C_TypeDef * I2Cx, uint32_t TypeAcknowledge)</code>
Function description	Prepare the generation of a ACKnowledge or Non ACKnowledge condition after the address receive match code or next received byte.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • TypeAcknowledge: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_I2C_ACK – LL_I2C_NACK
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Usage in Slave or Master mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 ACK LL_I2C_AcknowledgeNextData

LL_I2C_GenerateStartCondition

Function name	<code>__STATIC_INLINE void LL_I2C_GenerateStartCondition(I2C_TypeDef * I2Cx)</code>
Function description	Generate a START or RESTART condition.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • The START bit can be set even if bus is BUSY or I2C is in slave mode. This action has no effect when RELOAD is set.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 START LL_I2C_GenerateStartCondition

LL_I2C_GenerateStopCondition

Function name	<code>__STATIC_INLINE void LL_I2C_GenerateStopCondition(I2C_TypeDef * I2Cx)</code>
Function description	Generate a STOP condition after the current byte transfer (master mode).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 STOP LL_I2C_GenerateStopCondition

LL_I2C_EnableBitPOS

Function name	<code>_STATIC_INLINE void LL_I2C_EnableBitPOS (I2C_TypeDef * I2Cx)</code>
Function description	Enable bit POS (master/host mode).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • In that case, the ACK bit controls the (N)ACK of the next byte received or the PEC bit indicates that the next byte in shift register is a PEC.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 POS LL_I2C_EnableBitPOS

LL_I2C_DisableBitPOS

Function name	<code>_STATIC_INLINE void LL_I2C_DisableBitPOS (I2C_TypeDef * I2Cx)</code>
Function description	Disable bit POS (master/host mode).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • In that case, the ACK bit controls the (N)ACK of the current byte received or the PEC bit indicates that the current byte in shift register is a PEC.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 POS LL_I2C_DisableBitPOS

LL_I2C_IsEnabledBitPOS

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_IsEnabledBitPOS (I2C_TypeDef * I2Cx)</code>
Function description	Check if bit POS is enabled or disabled.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 POS LL_I2C_IsEnabledBitPOS

LL_I2C_GetTransferDirection

Function name	<code>_STATIC_INLINE uint32_t LL_I2C_GetTransferDirection (I2C_TypeDef * I2Cx)</code>
Function description	Indicate the value of transfer direction.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.

Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_I2C_DIRECTION_WRITE - LL_I2C_DIRECTION_READ
Notes	<ul style="list-style-type: none"> RESET: Bus is in read transfer (peripheral point of view). SET: Bus is in write transfer (peripheral point of view).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR2 TRA LL_I2C_GetTransferDirection

LL_I2C_EnableLastDMA

Function name	<code>__STATIC_INLINE void LL_I2C_EnableLastDMA (I2C_TypeDef * I2Cx)</code>
Function description	Enable DMA last transfer.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> This action mean that next DMA EOT is the last transfer.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 LAST LL_I2C_EnableLastDMA

LL_I2C_DisableLastDMA

Function name	<code>__STATIC_INLINE void LL_I2C_DisableLastDMA (I2C_TypeDef * I2Cx)</code>
Function description	Disable DMA last transfer.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> This action mean that next DMA EOT is not the last transfer.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 LAST LL_I2C_DisableLastDMA

LL_I2C_IsEnabledLastDMA

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledLastDMA (I2C_TypeDef * I2Cx)</code>
Function description	Check if DMA last transfer is enabled or disabled.
Parameters	<ul style="list-style-type: none"> I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 LAST LL_I2C_IsEnabledLastDMA

LL_I2C_EnableSMBusPECCompare

Function name	<code>__STATIC_INLINE void LL_I2C_EnableSMBusPECCompare(I2C_TypeDef * I2Cx)</code>
Function description	Enable transfer or internal comparison of the SMBus Packet Error byte (transmission or reception mode).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Macro <code>IS_SMBUS_ALL_INSTANCE(I2Cx)</code> can be used to check whether or not SMBus feature is supported by the I2Cx Instance. • This feature is cleared by hardware when the PEC byte is transferred or compared, or by a START or STOP condition, it is also cleared by software.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 PEC <code>LL_I2C_EnableSMBusPECCompare</code>

LL_I2C_DisableSMBusPECCompare

Function name	<code>__STATIC_INLINE void LL_I2C_DisableSMBusPECCompare(I2C_TypeDef * I2Cx)</code>
Function description	Disable transfer or internal comparison of the SMBus Packet Error byte (transmission or reception mode).
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Macro <code>IS_SMBUS_ALL_INSTANCE(I2Cx)</code> can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 PEC <code>LL_I2C_DisableSMBusPECCompare</code>

LL_I2C_IsEnabledSMBusPECCompare

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_IsEnabledSMBusPECCompare(I2C_TypeDef * I2Cx)</code>
Function description	Check if the SMBus Packet Error byte transfer or internal comparison is requested or not.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • Macro <code>IS_SMBUS_ALL_INSTANCE(I2Cx)</code> can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 PEC <code>LL_I2C_IsEnabledSMBusPECCompare</code>

LL_I2C_GetSMBusPEC

Function name	<code>__STATIC_INLINE uint32_t LL_I2C_GetSMBusPEC (I2C_TypeDef * I2Cx)</code>
Function description	Get the SMBus Packet Error byte calculated.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0xFF
Notes	<ul style="list-style-type: none"> • Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR2 PEC LL_I2C_GetSMBusPEC

LL_I2C_ReceiveData8

Function name	<code>__STATIC_INLINE uint8_t LL_I2C_ReceiveData8 (I2C_TypeDef * I2Cx)</code>
Function description	Read Receive Data register.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x0 and Max_Data=0xFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DR DR LL_I2C_ReceiveData8

LL_I2C_TransmitData8

Function name	<code>__STATIC_INLINE void LL_I2C_TransmitData8 (I2C_TypeDef * I2Cx, uint8_t Data)</code>
Function description	Write in Transmit Data Register .
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • Data: Value between Min_Data=0x0 and Max_Data=0xFF
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DR DR LL_I2C_TransmitData8

LL_I2C_Init

Function name	<code>uint32_t LL_I2C_Init (I2C_TypeDef * I2Cx, LL_I2C_InitTypeDef * I2C_InitStruct)</code>
Function description	Initialize the I2C registers according to the specified parameters in I2C_InitStruct.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance. • I2C_InitStruct: pointer to a LL_I2C_InitTypeDef structure.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value:

- SUCCESS: I2C registers are initialized
- ERROR: Not applicable

LL_I2C_DeInit

Function name	uint32_t LL_I2C_DeInit (I2C_TypeDef * I2Cx)
Function description	De-initialize the I2C registers to their default reset values.
Parameters	<ul style="list-style-type: none"> • I2Cx: I2C Instance.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> - SUCCESS: I2C registers are de-initialized - ERROR: I2C registers are not de-initialized

LL_I2C_StructInit

Function name	void LL_I2C_StructInit (LL_I2C_InitTypeDef * I2C_InitStruct)
Function description	Set each LL_I2C_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • I2C_InitStruct: Pointer to a LL_I2C_InitTypeDef structure.
Return values	<ul style="list-style-type: none"> • None:

57.3 I2C Firmware driver defines

57.3.1 I2C

Master Clock Speed Mode

LL_I2C_CLOCK_SPEED_STANDARD_MODE	Master clock speed range is standard mode
LL_I2C_CLOCK_SPEED_FAST_MODE	Master clock speed range is fast mode

Read Write Direction

LL_I2C_DIRECTION_WRITE	Bus is in write transfer
LL_I2C_DIRECTION_READ	Bus is in read transfer

Fast Mode Duty Cycle

LL_I2C_DUTYCYCLE_2	I2C fast mode Tlow/Thigh = 2
LL_I2C_DUTYCYCLE_16_9	I2C fast mode Tlow/Thigh = 16/9

Get Flags Defines

LL_I2C_SR1_SB	Start Bit (master mode)
LL_I2C_SR1_ADDR	Address sent (master mode) or Address matched flag (slave mode)
LL_I2C_SR1_BTF	Byte Transfer Finished flag
LL_I2C_SR1_ADD10	10-bit header sent (master mode)
LL_I2C_SR1_STOPF	Stop detection flag (slave mode)
LL_I2C_SR1_RXNE	Data register not empty (receivers)
LL_I2C_SR1_TXE	Data register empty (transmitters)

LL_I2C_SR1_BERR	Bus error
LL_I2C_SR1_ARLO	Arbitration lost
LL_I2C_SR1_AF	Acknowledge failure flag
LL_I2C_SR1_OVR	Overrun/Underrun
LL_I2C_SR1_PECERR	PEC Error in reception (SMBus mode)
LL_I2C_SR1_TIMEOUT	Timeout detection flag (SMBus mode)
LL_I2C_SR1_SMALERT	SMBus alert (SMBus mode)
LL_I2C_SR2_MSL	Master/Slave flag
LL_I2C_SR2_BUSY	Bus busy flag
LL_I2C_SR2_TRA	Transmitter/receiver direction
LL_I2C_SR2_GENCALL	General call address (Slave mode)
LL_I2C_SR2_SMBDEFAULT	SMBus Device default address (Slave mode)
LL_I2C_SR2_SMBHOST	SMBus Host address (Slave mode)
LL_I2C_SR2_DUALF	Dual flag (Slave mode)

Acknowledge Generation

LL_I2C_ACK	ACK is sent after current received byte.
LL_I2C_NACK	NACK is sent after current received byte.

IT Defines

LL_I2C_CR2_ITEVTEN	Events interrupts enable
LL_I2C_CR2_ITBUFEN	Buffer interrupts enable
LL_I2C_CR2_ITERREN	Error interrupts enable

Own Address 1 Length

LL_I2C_OWNADDRESS1_7BIT	Own address 1 is a 7-bit address.
LL_I2C_OWNADDRESS1_10BIT	Own address 1 is a 10-bit address.

Peripheral Mode

LL_I2C_MODE_I2C	I2C Master or Slave mode
LL_I2C_MODE_SMBUS_HOST	SMBus Host address acknowledge
LL_I2C_MODE_SMBUS_DEVICE	SMBus Device default mode (Default address not acknowledge)
LL_I2C_MODE_SMBUS_DEVICE_ARP	SMBus Device Default address acknowledge

Exported Macros Helper

<u>LL_I2C_FREQ_HZ_TO_MHZ</u>	Description:
	<ul style="list-style-type: none"> Convert Peripheral Clock Frequency in Mhz.
	Parameters: <ul style="list-style-type: none"> <u>PCLK</u>: This parameter must be a value of peripheral clock (in Hz).

[__LL_I2C_FREQ_MHZ_TO_HZ](#)**Return value:**

- Value: of peripheral clock (in Mhz)

Description:

- Convert Peripheral Clock Frequency in Hz.

Parameters:

- __PCLK__: This parameter must be a value of peripheral clock (in Mhz).

Return value:

- Value: of peripheral clock (in Hz)

[__LL_I2C_RISE_TIME](#)**Description:**

- Compute I2C Clock rising time.

Parameters:

- __FREQRANGE__: This parameter must be a value of peripheral clock (in Mhz).
- __SPEED__: This parameter must be a value lower than 400kHz (in Hz).

Return value:

- Value: between Min_Data=0x02 and Max_Data=0x3F

[__LL_I2C_SPEED_TO_CCR](#)**Description:**

- Compute Speed clock range to a Clock Control Register (I2C_CCR_CCR) value.

Parameters:

- __PCLK__: This parameter must be a value of peripheral clock (in Hz).
- __SPEED__: This parameter must be a value lower than 400kHz (in Hz).
- __DUTYCYCLE__: This parameter can be one of the following values:
 - LL_I2C_DUTYCYCLE_2
 - LL_I2C_DUTYCYCLE_16_9

Return value:

- Value: between Min_Data=0x004 and Max_Data=0xFFFF, except in FAST DUTY mode where Min_Data=0x001.

[__LL_I2C_SPEED_STANDARD_TO_CCR](#)**Description:**

- Compute Speed Standard clock range to a Clock Control Register (I2C_CCR_CCR) value.

Parameters:

- __PCLK__: This parameter must be a

- SPEED: This parameter must be a value lower than 100kHz (in Hz).

Return value:

- Value: between Min_Data=0x004 and Max_Data=0xFFFF.

_LL_I2C_SPEED_FAST_TO_CCR**Description:**

- Compute Speed Fast clock range to a Clock Control Register (I2C_CCR_CCR) value.

Parameters:

- PCLK: This parameter must be a value of peripheral clock (in Hz).
- SPEED: This parameter must be a value between Min_Data=100Khz and Max_Data=400Khz (in Hz).
- DUTYCYCLE: This parameter can be one of the following values:
 - LL_I2C_DUTYCYCLE_2
 - LL_I2C_DUTYCYCLE_16_9

Return value:

- Value: between Min_Data=0x001 and Max_Data=0xFFFF

_LL_I2C_10BIT_ADDRESS**Description:**

- Get the Least significant bits of a 10-Bits address.

Parameters:

- ADDRESS: This parameter must be a value of a 10-Bits slave address.

Return value:

- Value: between Min_Data=0x00 and Max_Data=0xFF

_LL_I2C_10BIT_HEADER_WRITE**Description:**

- Convert a 10-Bits address to a 10-Bits header with Write direction.

Parameters:

- ADDRESS: This parameter must be a value of a 10-Bits slave address.

Return value:

- Value: between Min_Data=0xF0 and Max_Data=0xF6

_LL_I2C_10BIT_HEADER_READ**Description:**

- Convert a 10-Bits address to a 10-Bits

header with Read direction.

Parameters:

- __ADDRESS__: This parameter must be a value of a 10-Bits slave address.

Return value:

- Value: between Min_Data=0xF1 and Max_Data=0xF7

Common Write and read registers Macros

`LL_I2C_WriteReg`

Description:

- Write a value in I2C register.

Parameters:

- __INSTANCE__: I2C Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

`LL_I2C_ReadReg`

Description:

- Read a value in I2C register.

Parameters:

- __INSTANCE__: I2C Instance
- __REG__: Register to be read

Return value:

- Register: value

58 LL I2S Generic Driver

58.1 I2S Firmware driver registers structures

58.1.1 LL_I2S_InitTypeDef

Data Fields

- *uint32_t Mode*
- *uint32_t Standard*
- *uint32_t DataFormat*
- *uint32_t MCLKOutput*
- *uint32_t AudioFreq*
- *uint32_t ClockPolarity*

Field Documentation

- ***uint32_t LL_I2S_InitTypeDef::Mode***
Specifies the I2S operating mode. This parameter can be a value of **I2S_LL_EC_MODE**This feature can be modified afterwards using unitary function **LL_I2S_SetTransferMode()**.
- ***uint32_t LL_I2S_InitTypeDef::Standard***
Specifies the standard used for the I2S communication. This parameter can be a value of **I2S_LL_EC_STANDARD**This feature can be modified afterwards using unitary function **LL_I2S_SetStandard()**.
- ***uint32_t LL_I2S_InitTypeDef::DataFormat***
Specifies the data format for the I2S communication. This parameter can be a value of **I2S_LL_EC_DATA_FORMAT**This feature can be modified afterwards using unitary function **LL_I2S_SetDataFormat()**.
- ***uint32_t LL_I2S_InitTypeDef::MCLKOutput***
Specifies whether the I2S MCLK output is enabled or not. This parameter can be a value of **I2S_LL_EC_MCLK_OUTPUT**This feature can be modified afterwards using unitary functions **LL_I2S_EnableMasterClock()** or **LL_I2S_DisableMasterClock()**.
- ***uint32_t LL_I2S_InitTypeDef::AudioFreq***
Specifies the frequency selected for the I2S communication. This parameter can be a value of **I2S_LL_EC_AUDIO_FREQ**Audio Frequency can be modified afterwards using Reference manual formulas to calculate Prescaler Linear, Parity and unitary functions **LL_I2S_SetPrescalerLinear()** and **LL_I2S_SetPrescalerParity()** to set it.
- ***uint32_t LL_I2S_InitTypeDef::ClockPolarity***
Specifies the idle state of the I2S clock. This parameter can be a value of **I2S_LL_EC_POLARITY**This feature can be modified afterwards using unitary function **LL_I2S_SetClockPolarity()**.

58.2 I2S Firmware driver API description

58.2.1 Detailed description of functions

LL_I2S_Enable

Function name	<code>__STATIC_INLINE void LL_I2S_Enable (SPI_TypeDef * SPIx)</code>
Function description	Select I2S mode and Enable I2S peripheral.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance

Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR I2SMOD LL_I2S_Enable • I2SCFGR I2SE LL_I2S_Enable

LL_I2S_Disable

Function name	<code>__STATIC_INLINE void LL_I2S_Disable (SPI_TypeDef * SPIx)</code>
Function description	Disable I2S peripheral.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR I2SE LL_I2S_Disable

LL_I2S_IsEnabled

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsEnabled (SPI_TypeDef * SPIx)</code>
Function description	Check if I2S peripheral is enabled.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR I2SE LL_I2S_IsEnabled

LL_I2S_SetDataFormat

Function name	<code>__STATIC_INLINE void LL_I2S_SetDataFormat (SPI_TypeDef * SPIx, uint32_t DataFormat)</code>
Function description	Set I2S data frame length.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • DataFormat: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_I2S_DATAFORMAT_16B – LL_I2S_DATAFORMAT_16B_EXTENDED – LL_I2S_DATAFORMAT_24B – LL_I2S_DATAFORMAT_32B
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR DATLEN LL_I2S_SetDataFormat • I2SCFGR CHLEN LL_I2S_SetDataFormat

LL_I2S_GetDataFormat

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_GetDataFormat (SPI_TypeDef * SPIx)</code>
---------------	--

Function description	Get I2S data frame length.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_DATAFORMAT_16B - LL_I2S_DATAFORMAT_16B_EXTENDED - LL_I2S_DATAFORMAT_24B - LL_I2S_DATAFORMAT_32B
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR DATLEN LL_I2S_GetDataFormat • I2SCFGR CHLEN LL_I2S_GetDataFormat

LL_I2S_SetClockPolarity

Function name	<code>_STATIC_INLINE void LL_I2S_SetClockPolarity(SPI_TypeDef * SPIx, uint32_t ClockPolarity)</code>
Function description	Set I2S clock polarity.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • ClockPolarity: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_POLARITY_LOW - LL_I2S_POLARITY_HIGH
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR CKPOL LL_I2S_SetClockPolarity

LL_I2S_GetClockPolarity

Function name	<code>_STATIC_INLINE uint32_t LL_I2S_GetClockPolarity(SPI_TypeDef * SPIx)</code>
Function description	Get I2S clock polarity.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_POLARITY_LOW - LL_I2S_POLARITY_HIGH
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR CKPOL LL_I2S_GetClockPolarity

LL_I2S_SetStandard

Function name	<code>_STATIC_INLINE void LL_I2S_SetStandard (SPI_TypeDef * SPIx, uint32_t Standard)</code>
Function description	Set I2S standard protocol.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • Standard: This parameter can be one of the following values:

	<ul style="list-style-type: none"> - LL_I2S_STANDARD_PHILIPS - LL_I2S_STANDARD_MSB - LL_I2S_STANDARD_LSB - LL_I2S_STANDARD_PCM_SHORT - LL_I2S_STANDARD_PCM_LONG
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR I2SSTD LL_I2S_SetStandard • I2SCFGR PCMSYNC LL_I2S_SetStandard

LL_I2S_GetStandard

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_GetStandard (SPI_TypeDef * SPIx)</code>
Function description	Get I2S standard protocol.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_STANDARD_PHILIPS - LL_I2S_STANDARD_MSB - LL_I2S_STANDARD_LSB - LL_I2S_STANDARD_PCM_SHORT - LL_I2S_STANDARD_PCM_LONG
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR I2SSTD LL_I2S_SetStandard • I2SCFGR PCMSYNC LL_I2S_SetStandard

LL_I2S_SetTransferMode

Function name	<code>__STATIC_INLINE void LL_I2S_SetTransferMode (SPI_TypeDef * SPIx, uint32_t Mode)</code>
Function description	Set I2S transfer mode.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • Mode: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_MODE_SLAVE_TX - LL_I2S_MODE_SLAVE_RX - LL_I2S_MODE_MASTER_TX - LL_I2S_MODE_MASTER_RX
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • I2SCFGR I2SCFG LL_I2S_SetTransferMode

LL_I2S_GetTransferMode

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_GetTransferMode (SPI_TypeDef * SPIx)</code>
Function description	Get I2S transfer mode.

Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_MODE_SLAVE_TX - LL_I2S_MODE_SLAVE_RX - LL_I2S_MODE_MASTER_TX - LL_I2S_MODE_MASTER_RX
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> I2SCFGR I2SCFG LL_I2S_GetTransferMode

LL_I2S_SetPrescalerLinear

Function name	<code>__STATIC_INLINE void LL_I2S_SetPrescalerLinear(SPI_TypeDef * SPIx, uint8_t PrescalerLinear)</code>
Function description	Set I2S linear prescaler.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance PrescalerLinear: Value between Min_Data=0x02 and Max_Data=0xFF
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> I2SPR I2SDIV LL_I2S_SetPrescalerLinear

LL_I2S_GetPrescalerLinear

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_GetPrescalerLinear(SPI_TypeDef * SPIx)</code>
Function description	Get I2S linear prescaler.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> PrescalerLinear: Value between Min_Data=0x02 and Max_Data=0xFF

Reference Manual to
LL API cross
reference:

- I2SPR I2SDIV LL_I2S_GetPrescalerLinear

LL_I2S_SetPrescalerParity

Function name	<code>__STATIC_INLINE void LL_I2S_SetPrescalerParity(SPI_TypeDef * SPIx, uint32_t PrescalerParity)</code>
Function description	Set I2S parity prescaler.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance PrescalerParity: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_PRESCALER_PARITY_EVEN - LL_I2S_PRESCALER_PARITY_ODD
Return values	<ul style="list-style-type: none"> None:

- Reference Manual to I2SPR ODD LL_I2S_SetPrescalerParity
LL API cross reference:

LL_I2S_GetPrescalerParity

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_GetPrescalerParity(SPI_TypeDef * SPIx)</code>
Function description	Get I2S parity prescaler.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_PRESCALER_PARITY EVEN - LL_I2S_PRESCALER_PARITY ODD
Reference Manual to I2SPR ODD LL_I2S_SetPrescalerParity LL API cross reference:	<ul style="list-style-type: none"> • I2SPR ODD LL_I2S_SetPrescalerParity

LL_I2S_EnableMasterClock

Function name	<code>__STATIC_INLINE void LL_I2S_EnableMasterClock(SPI_TypeDef * SPIx)</code>
Function description	Enable the master clock output (Pin MCK)
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to I2SPR MCKOE LL_I2S_EnableMasterClock LL API cross reference:	<ul style="list-style-type: none"> • I2SPR MCKOE LL_I2S_EnableMasterClock

LL_I2S_DisableMasterClock

Function name	<code>__STATIC_INLINE void LL_I2S_DisableMasterClock(SPI_TypeDef * SPIx)</code>
Function description	Disable the master clock output (Pin MCK)
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to I2SPR MCKOE LL_I2S_DisableMasterClock LL API cross reference:	<ul style="list-style-type: none"> • I2SPR MCKOE LL_I2S_DisableMasterClock

LL_I2S_IsEnabledMasterClock

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsEnabledMasterClock(SPI_TypeDef * SPIx)</code>
Function description	Check if the master clock output (Pin MCK) is enabled.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- I2SPR MCKOE LL_I2S_IsEnabledMasterClock

LL_I2S_IsActiveFlag_RXNE

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_RXNE(SPI_TypeDef * SPIx)</code>
Function description	Check if Rx buffer is not empty.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- SR RXNE LL_I2S_IsActiveFlag_RXNE

LL_I2S_IsActiveFlag_TXE

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_TXE(SPI_TypeDef * SPIx)</code>
Function description	Check if Tx buffer is empty.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- SR TXE LL_I2S_IsActiveFlag_TXE

LL_I2S_IsActiveFlag_BSY

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_BSY(SPI_TypeDef * SPIx)</code>
Function description	Get busy flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- SR BSY LL_I2S_IsActiveFlag_BSY

LL_I2S_IsActiveFlag_OVR

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_OVR(SPI_TypeDef * SPIx)</code>
Function description	Get overrun error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
- SR OVR LL_I2S_IsActiveFlag_OVR

reference:

LL_I2S_IsActiveFlag_UDR

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_UDR (SPI_TypeDef * SPIx)</code>
Function description	Get underrun error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR UDR LL_I2S_IsActiveFlag_UDR

LL_I2S_IsActiveFlag_FRE

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_FRE (SPI_TypeDef * SPIx)</code>
Function description	Get frame format error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR FRE LL_I2S_IsActiveFlag_FRE

LL_I2S_IsActiveFlag_CHSIDE

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_CHSIDE (SPI_TypeDef * SPIx)</code>
Function description	Get channel side flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> • 0: Channel Left has to be transmitted or has been received 1: Channel Right has to be transmitted or has been received It has no significance in PCM mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR CHSIDE LL_I2S_IsActiveFlag_CHSIDE

LL_I2S_ClearFlag_OVR

Function name	<code>__STATIC_INLINE void LL_I2S_ClearFlag_OVR (SPI_TypeDef * SPIx)</code>
Function description	Clear overrun error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:

- Reference Manual to
LL API cross
reference:
- SR OVR LL_I2S_ClearFlag_OVR

LL_I2S_ClearFlag_UDR

Function name	<code>_STATIC_INLINE void LL_I2S_ClearFlag_UDR (SPI_TypeDef * SPIx)</code>
Function description	Clear underrun error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:

Reference Manual to
LL API cross
reference:

- SR UDR LL_I2S_ClearFlag_UDR

LL_I2S_ClearFlag_FRE

Function name	<code>_STATIC_INLINE void LL_I2S_ClearFlag_FRE (SPI_TypeDef * SPIx)</code>
Function description	Clear frame format error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:

Reference Manual to
LL API cross
reference:

- SR FRE LL_I2S_ClearFlag_FRE

LL_I2S_EnableIT_ERR

Function name	<code>_STATIC_INLINE void LL_I2S_EnableIT_ERR (SPI_TypeDef * SPIx)</code>
Function description	Enable error IT.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit controls the generation of an interrupt when an error condition occurs (OVR, UDR and FRE in I2S mode).

Reference Manual to
LL API cross
reference:

- CR2 ERRIE LL_I2S_EnableIT_ERR

LL_I2S_EnableIT_RXNE

Function name	<code>_STATIC_INLINE void LL_I2S_EnableIT_RXNE (SPI_TypeDef * SPIx)</code>
Function description	Enable Rx buffer not empty IT.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance

Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXNEIE LL_I2S_EnableIT_RXNE

LL_I2S_EnableIT_TXE

Function name	<code>__STATIC_INLINE void LL_I2S_EnableIT_TXE (SPI_TypeDef * SPIx)</code>
Function description	Enable Tx buffer empty IT.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 TXEIE LL_I2S_EnableIT_TXE

LL_I2S_DisableIT_ERR

Function name	<code>__STATIC_INLINE void LL_I2S_DisableIT_ERR (SPI_TypeDef * SPIx)</code>
Function description	Disable error IT.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> This bit controls the generation of an interrupt when an error condition occurs (OVR, UDR and FRE in I2S mode).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ERRIE LL_I2S_DisableIT_ERR

LL_I2S_DisableIT_RXNE

Function name	<code>__STATIC_INLINE void LL_I2S_DisableIT_RXNE (SPI_TypeDef * SPIx)</code>
Function description	Disable Rx buffer not empty IT.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXNEIE LL_I2S_DisableIT_RXNE

LL_I2S_DisableIT_TXE

Function name	<code>__STATIC_INLINE void LL_I2S_DisableIT_TXE (SPI_TypeDef * SPIx)</code>
Function description	Disable Tx buffer empty IT.

Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 TXEIE LL_I2S_DisableIT_TXE

LL_I2S_IsEnabledIT_ERR

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsEnabledIT_ERR(SPI_TypeDef * SPIx)</code>
Function description	Check if ERR IT is enabled.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ERRIE LL_I2S_IsEnabledIT_ERR

LL_I2S_IsEnabledIT_RXNE

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsEnabledIT_RXNE(SPI_TypeDef * SPIx)</code>
Function description	Check if RXNE IT is enabled.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXNEIE LL_I2S_IsEnabledIT_RXNE

LL_I2S_IsEnabledIT_TXE

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsEnabledIT_TXE(SPI_TypeDef * SPIx)</code>
Function description	Check if TXE IT is enabled.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 TXEIE LL_I2S_IsEnabledIT_TXE

LL_I2S_EnableDMAReq_RX

Function name	<code>__STATIC_INLINE void LL_I2S_EnableDMAReq_RX(SPI_TypeDef * SPIx)</code>
Function description	Enable DMA Rx.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance

Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXDMAEN LL_I2S_EnableDMAReq_RX

LL_I2S_DisableDMAReq_RX

Function name	<code>__STATIC_INLINE void LL_I2S_DisableDMAReq_RX(SPI_TypeDef * SPIx)</code>
Function description	Disable DMA Rx.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXDMAEN LL_I2S_DisableDMAReq_RX

LL_I2S_IsEnabledDMAReq_RX

Function name	<code>__STATIC_INLINE uint32_t LL_I2S_IsEnabledDMAReq_RX(SPI_TypeDef * SPIx)</code>
Function description	Check if DMA Rx is enabled.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXDMAEN LL_I2S_IsEnabledDMAReq_RX

LL_I2S_EnableDMAReq_TX

Function name	<code>__STATIC_INLINE void LL_I2S_EnableDMAReq_TX(SPI_TypeDef * SPIx)</code>
Function description	Enable DMA Tx.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 TXDMAEN LL_I2S_EnableDMAReq_TX

LL_I2S_DisableDMAReq_TX

Function name	<code>__STATIC_INLINE void LL_I2S_DisableDMAReq_TX(SPI_TypeDef * SPIx)</code>
Function description	Disable DMA Tx.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:

- Reference Manual to
LL API cross
reference:
- CR2 TXDMAEN LL_I2S_DisableDMAReq_TX

LL_I2S_IsEnabledDMAReq_TX

Function name	<code>_STATIC_INLINE uint32_t LL_I2S_IsEnabledDMAReq_TX(SPI_TypeDef * SPIx)</code>
Function description	Check if DMA Tx is enabled.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- CR2 TXDMAEN LL_I2S_IsEnabledDMAReq_TX

LL_I2S_ReceiveData16

Function name	<code>_STATIC_INLINE uint16_t LL_I2S_ReceiveData16(SPI_TypeDef * SPIx)</code>
Function description	Read 16-Bits in data register.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • RxData: Value between Min_Data=0x0000 and Max_Data=0xFFFF

- Reference Manual to
LL API cross
reference:
- DR DR LL_I2S_ReceiveData16

LL_I2S_TransmitData16

Function name	<code>_STATIC_INLINE void LL_I2S_TransmitData16 (SPI_TypeDef * SPIx, uint16_t TxData)</code>
Function description	Write 16-Bits in data register.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • TxData: Value between Min_Data=0x0000 and Max_Data=0xFFFF
Return values	<ul style="list-style-type: none"> • None:

- Reference Manual to
LL API cross
reference:
- DR DR LL_I2S_TransmitData16

LL_I2S_DelInit

Function name	<code>ErrorStatus LL_I2S_DelInit (SPI_TypeDef * SPIx)</code>
Function description	De-initialize the SPI/I2S registers to their default reset values.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value:

- SUCCESS: SPI registers are de-initialized
- ERROR: SPI registers are not de-initialized

LL_I2S_Init

Function name	ErrorStatus LL_I2S_Init (SPI_TypeDef * SPIx, LL_I2S_InitTypeDef * I2S_InitStruct)
Function description	Initializes the SPI/I2S registers according to the specified parameters in I2S_InitStruct.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • I2S_InitStruct: pointer to a LL_I2S_InitTypeDef structure
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> - SUCCESS: SPI registers are Initialized - ERROR: SPI registers are not Initialized
Notes	<ul style="list-style-type: none"> • As some bits in SPI configuration registers can only be written when the SPI is disabled (SPI_CR1_SPE bit =0), SPI IP should be in disabled state prior calling this function. Otherwise, ERROR result will be returned.

LL_I2S_StructInit

Function name	void LL_I2S_StructInit (LL_I2S_InitTypeDef * I2S_InitStruct)
Function description	Set each LL_I2S_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • I2S_InitStruct: pointer to a LL_I2S_InitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> • None:

LL_I2S_ConfigPrescaler

Function name	void LL_I2S_ConfigPrescaler (SPI_TypeDef * SPIx, uint32_t PrescalerLinear, uint32_t PrescalerParity)
Function description	Set linear and parity prescaler.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • PrescalerLinear: value: Min_Data=0x02 and Max_Data=0xFF. • PrescalerParity: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_I2S_PRESCALER_PARITY_EVEN - LL_I2S_PRESCALER_PARITY_ODD
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • To calculate value of PrescalerLinear(I2SDIV[7:0] bits) and PrescalerParity(ODD bit) Check Audio frequency table and formulas inside Reference Manual (SPI/I2S).

58.3 I2S Firmware driver defines

58.3.1 I2S

Audio Frequency

LL_I2S_AUDIOFREQ_192K	Audio Frequency configuration 192000 Hz
LL_I2S_AUDIOFREQ_96K	Audio Frequency configuration 96000 Hz
LL_I2S_AUDIOFREQ_48K	Audio Frequency configuration 48000 Hz
LL_I2S_AUDIOFREQ_44K	Audio Frequency configuration 44100 Hz
LL_I2S_AUDIOFREQ_32K	Audio Frequency configuration 32000 Hz
LL_I2S_AUDIOFREQ_22K	Audio Frequency configuration 22050 Hz
LL_I2S_AUDIOFREQ_16K	Audio Frequency configuration 16000 Hz
LL_I2S_AUDIOFREQ_11K	Audio Frequency configuration 11025 Hz
LL_I2S_AUDIOFREQ_8K	Audio Frequency configuration 8000 Hz
LL_I2S_AUDIOFREQ_DEFAULT	Audio Freq not specified. Register I2SDIV = 2

Data format

LL_I2S_DATAFORMAT_16B	Data length 16 bits, Channel lenght 16bit
LL_I2S_DATAFORMAT_16B_EXTENDED	Data length 16 bits, Channel lenght 32bit
LL_I2S_DATAFORMAT_24B	Data length 24 bits, Channel lenght 32bit
LL_I2S_DATAFORMAT_32B	Data length 16 bits, Channel lenght 32bit

Get Flags Defines

LL_I2S_SR_RXNE	Rx buffer not empty flag
LL_I2S_SR_TXE	Tx buffer empty flag
LL_I2S_SR_BSY	Busy flag
LL_I2S_SR_UDR	Underrun flag
LL_I2S_SR_OVR	Overrun flag
LL_I2S_SR_FRE	TI mode frame format error flag

MCLK Output

LL_I2S_MCLK_OUTPUT_DISABLE	Master clock output is disabled
LL_I2S_MCLK_OUTPUT_ENABLE	Master clock output is enabled

Operation Mode

LL_I2S_MODE_SLAVE_TX	Slave Tx configuration
LL_I2S_MODE_SLAVE_RX	Slave Rx configuration
LL_I2S_MODE_MASTER_TX	Master Tx configuration
LL_I2S_MODE_MASTER_RX	Master Rx configuration

Clock Polarity

LL_I2S_POLARITY_LOW	Clock steady state is low level
LL_I2S_POLARITY_HIGH	Clock steady state is high level

Prescaler Factor

LL_I2S_PRESCALER_PARITY_EVEN	Odd factor: Real divider value is = I2SDIV * 2
LL_I2S_PRESCALER_PARITY_ODD	Odd factor: Real divider value is = (I2SDIV * 2)+1

I2s Standard

LL_I2S_STANDARD_PHILIPS	I2S standard philips
LL_I2S_STANDARD_MSB	MSB justified standard (left justified)
LL_I2S_STANDARD_LSB	LSB justified standard (right justified)
LL_I2S_STANDARD_PCM_SHORT	PCM standard, short frame synchronization
LL_I2S_STANDARD_PCM_LONG	PCM standard, long frame synchronization

Common Write and read registers Macros

LL_I2S_WriteReg **Description:**

- Write a value in I2S register.

Parameters:

- **_INSTANCE_**: I2S Instance
- **_REG_**: Register to be written
- **_VALUE_**: Value to be written in the register

Return value:

- None

LL_I2S_ReadReg **Description:**

- Read a value in I2S register.

Parameters:

- **_INSTANCE_**: I2S Instance
- **_REG_**: Register to be read

Return value:

- Register: value

59 LL IWDG Generic Driver

59.1 IWDG Firmware driver API description

59.1.1 Detailed description of functions

LL_IWDG_Enable

Function name **`__STATIC_INLINE void LL_IWDG_Enable (IWDG_TypeDef * IWDGx)`**

Function description Start the Independent Watchdog.

- **IWDGx:** IWDG Instance

Return values **None:**

Notes Except if the hardware watchdog option is selected

Reference Manual to **KR KEY LL_IWDG_Enable**

LL API cross reference:

LL_IWDG_ReloadCounter

Function name **`__STATIC_INLINE void LL_IWDG_ReloadCounter (IWDG_TypeDef * IWDGx)`**

Function description Reloads IWDG counter with value defined in the reload register.

- **IWDGx:** IWDG Instance

Return values **None:**

Reference Manual to **KR KEY LL_IWDG_ReloadCounter**

LL API cross reference:

LL_IWDG_EnableWriteAccess

Function name **`__STATIC_INLINE void LL_IWDG_EnableWriteAccess (IWDG_TypeDef * IWDGx)`**

Function description Enable write access to IWDG_PR, IWDG_RLR and IWDG_WINR registers.

- **IWDGx:** IWDG Instance

Return values **None:**

Reference Manual to **KR KEY LL_IWDG_EnableWriteAccess**

LL API cross reference:

LL_IWDG_DisableWriteAccess

Function name **`__STATIC_INLINE void LL_IWDG_DisableWriteAccess`**

(IWDG_TypeDef * IWDGx)

Function description	Disable write access to IWDG_PR, IWDG_RLR and IWDG_WINR registers.
Parameters	<ul style="list-style-type: none"> • IWDGx: IWDG Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • KR KEY LL_IWDG_DisableWriteAccess

LL_IWDG_SetPrescaler

Function name **__STATIC_INLINE void LL_IWDG_SetPrescaler
(IWDG_TypeDef * IWDGx, uint32_t Prescaler)**

Function description	Select the prescaler of the IWDG.
Parameters	<ul style="list-style-type: none"> • IWDGx: IWDG Instance • Prescaler: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_IWDG_PRESCALER_4 - LL_IWDG_PRESCALER_8 - LL_IWDG_PRESCALER_16 - LL_IWDG_PRESCALER_32 - LL_IWDG_PRESCALER_64 - LL_IWDG_PRESCALER_128 - LL_IWDG_PRESCALER_256
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • PR PR LL_IWDG_SetPrescaler

LL_IWDG_GetPrescaler

Function name **__STATIC_INLINE uint32_t LL_IWDG_GetPrescaler
(IWDG_TypeDef * IWDGx)**

Function description	Get the selected prescaler of the IWDG.
Parameters	<ul style="list-style-type: none"> • IWDGx: IWDG Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_IWDG_PRESCALER_4 - LL_IWDG_PRESCALER_8 - LL_IWDG_PRESCALER_16 - LL_IWDG_PRESCALER_32 - LL_IWDG_PRESCALER_64 - LL_IWDG_PRESCALER_128 - LL_IWDG_PRESCALER_256
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • PR PR LL_IWDG_GetPrescaler

LL_IWDG_SetReloadCounter

Function name	<code>_STATIC_INLINE void LL_IWDG_SetReloadCounter (IWDG_TypeDef * IWDGx, uint32_t Counter)</code>
Function description	Specify the IWDG down-counter reload value.
Parameters	<ul style="list-style-type: none"> • IWDGx: IWDG Instance • Counter: Value between Min_Data=0 and Max_Data=0x0FFF
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RLR RL LL_IWDG_SetReloadCounter

LL_IWDG_GetReloadCounter

Function name	<code>_STATIC_INLINE uint32_t LL_IWDG_GetReloadCounter (IWDG_TypeDef * IWDGx)</code>
Function description	Get the specified IWDG down-counter reload value.
Parameters	<ul style="list-style-type: none"> • IWDGx: IWDG Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0 and Max_Data=0x0FFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RLR RL LL_IWDG_GetReloadCounter

LL_IWDG_IsActiveFlag_PVU

Function name	<code>_STATIC_INLINE uint32_t LL_IWDG_IsActiveFlag_PVU (IWDG_TypeDef * IWDGx)</code>
Function description	Check if flag Prescaler Value Update is set or not.
Parameters	<ul style="list-style-type: none"> • IWDGx: IWDG Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR PVU LL_IWDG_IsActiveFlag_PVU

LL_IWDG_IsActiveFlag_RVU

Function name	<code>_STATIC_INLINE uint32_t LL_IWDG_IsActiveFlag_RVU (IWDG_TypeDef * IWDGx)</code>
Function description	Check if flag Reload Value Update is set or not.
Parameters	<ul style="list-style-type: none"> • IWDGx: IWDG Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR RVU LL_IWDG_IsActiveFlag_RVU

LL_IWDG_IsReady

Function name **_STATIC_INLINE uint32_t LL_IWDG_IsReady
(IWDG_TypeDef * IWDGx)**

Function description Check if all flags Prescaler, Reload & Window Value Update are reset or not.

Parameters • **IWDGx:** IWDG Instance

Return values • **State:** of bits (1 or 0).

Reference Manual to
LL API cross
reference:
• SR PVU LL_IWDG_IsReady
• SR RVU LL_IWDG_IsReady

59.2 IWDG Firmware driver defines

59.2.1 IWDG

Get Flags Defines

LL_IWDG_SR_PVU Watchdog prescaler value update

LL_IWDG_SR_RVU Watchdog counter reload value update

Prescaler Divider

LL_IWDG_PRESCALER_4 Divider by 4

LL_IWDG_PRESCALER_8 Divider by 8

LL_IWDG_PRESCALER_16 Divider by 16

LL_IWDG_PRESCALER_32 Divider by 32

LL_IWDG_PRESCALER_64 Divider by 64

LL_IWDG_PRESCALER_128 Divider by 128

LL_IWDG_PRESCALER_256 Divider by 256

Common Write and read registers Macros

LL_IWDG_WriteReg **Description:**

- Write a value in IWDG register.

Parameters:

- **_INSTANCE_**: IWDG Instance
- **_REG_**: Register to be written
- **_VALUE_**: Value to be written in the register

Return value:

- None

LL_IWDG_ReadReg **Description:**

- Read a value in IWDG register.

Parameters:

- **_INSTANCE_**: IWDG Instance
- **_REG_**: Register to be read

Return value:

- Register: value

60 LL OPAMP Generic Driver

60.1 OPAMP Firmware driver registers structures

60.1.1 LL_OPAMP_InitTypeDef

Data Fields

- *uint32_t PowerMode*
- *uint32_t FunctionalMode*
- *uint32_t InputNonInverting*
- *uint32_t InputInverting*

Field Documentation

- ***uint32_t LL_OPAMP_InitTypeDef::PowerMode***
Set OPAMP power mode. This parameter can be a value of
OPAMP_LL_EC_POWERMODEThis feature can be modified afterwards using unitary function **LL_OPAMP_SetPowerMode()**.
- ***uint32_t LL_OPAMP_InitTypeDef::FunctionalMode***
Set OPAMP functional mode by setting internal connections: OPAMP operation in standalone, follower, ... This parameter can be a value of
OPAMP_LL_EC_FUNCTIONAL_MODEThis feature can be modified afterwards using unitary function **LL_OPAMP_SetFunctionalMode()**.
- ***uint32_t LL_OPAMP_InitTypeDef::InputNonInverting***
Set OPAMP input non-inverting connection. This parameter can be a value of
OPAMP_LL_EC_INPUT_NONINVERTINGThis feature can be modified afterwards using unitary function **LL_OPAMP_SetInputNonInverting()**.
- ***uint32_t LL_OPAMP_InitTypeDef::InputInverting***
Set OPAMP inverting input connection. This parameter can be a value of
OPAMP_LL_EC_INPUT_INVERTING
Note:OPAMP inverting input is used with OPAMP in mode standalone. Otherwise (OPAMP in mode follower), OPAMP inverting input is not used (not connected to GPIO pin), this parameter is discarded. This feature can be modified afterwards using unitary function **LL_OPAMP_SetInputInverting()**.

60.2 OPAMP Firmware driver API description

60.2.1 Detailed description of functions

LL_OPAMP_SetCommonPowerRange

Function name ***_STATIC_INLINE void LL_OPAMP_SetCommonPowerRange (OPAMP_Common_TypeDef * OPAMPxy_COMMON, uint32_t PowerRange)***

Function description Set OPAMP power range.

- Parameters
- ***OPAMPxy_COMMON:*** OPAMP common instance (can be set directly from CMSIS definition or by using helper macro ***_LL_OPAMP_COMMON_INSTANCE()***)
 - ***PowerRange:*** This parameter can be one of the following values:
 - ***LL_OPAMP_POWERSUPPLY_RANGE_LOW***

	– LL_OPAMP_POWERSUPPLY_RANGE_HIGH
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> The OPAMP power range applies to several OPAMP instances (if several OPAMP instances available on the selected device). On this STM32 serie, setting of this feature is conditioned to OPAMP state: All OPAMP instances of the OPAMP common group must be disabled. This check can be done with function LL_OPAMP_IsEnabled() for each OPAMP instance or by using helper macro __LL_OPAMP_IS_ENABLED_ALL_COMMON_INSTANCE().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR AOP_RANGE LL_OPAMP_SetCommonPowerRange

LL_OPAMP_GetCommonPowerRange

Function name	<code>__STATIC_INLINE uint32_t LL_OPAMP_GetCommonPowerRange (OPAMP_Common_TypeDef * OPAMPxy_COMMON)</code>
Function description	Get OPAMP power range.
Parameters	<ul style="list-style-type: none"> OPAMPxy_COMMON: OPAMP common instance (can be set directly from CMSIS definition or by using helper macro __LL_OPAMP_COMMON_INSTANCE())
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_OPAMP_POWERSUPPLY_RANGE_LOW – LL_OPAMP_POWERSUPPLY_RANGE_HIGH
Notes	<ul style="list-style-type: none"> The OPAMP power range applies to several OPAMP instances (if several OPAMP instances available on the selected device).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR AOP_RANGE LL_OPAMP_GetCommonPowerRange

LL_OPAMP_SetPowerMode

Function name	<code>__STATIC_INLINE void LL_OPAMP_SetPowerMode (OPAMP_TypeDef * OPAMPx, uint32_t PowerMode)</code>
Function description	Set OPAMP power mode.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance PowerMode: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_OPAMP_POWERMODE_NORMAL – LL_OPAMP_POWERMODE_LOWPOWER
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> The OPAMP must be disabled to change this configuration.
Reference Manual to LL API cross	<ul style="list-style-type: none"> CSR OPA1LPM LL_OPAMP_SetPowerMode

- reference:
- CSR OPA2LPM LL_OPAMP_SetPowerMode
 - CSR OPA3LPM LL_OPAMP_SetPowerMode

LL_OPAMP_GetPowerMode

Function name **`_STATIC_INLINE uint32_t LL_OPAMP_GetPowerMode(
OPAMP_TypeDef * OPAMPx)`**

Function description Get OPAMP power mode.

Parameters • **OPAMPx:** OPAMP instance

Return values • **Returned:** value can be one of the following values:

- LL_OPAMP_POWERMODE_NORMAL
- LL_OPAMP_POWERMODE_LOWPOWER

Reference Manual to
LL API cross
reference:

- CSR OPA1LPM LL_OPAMP_GetPowerMode
- CSR OPA2LPM LL_OPAMP_GetPowerMode
- CSR OPA3LPM LL_OPAMP_GetPowerMode

LL_OPAMP_SetMode

Function name **`_STATIC_INLINE void LL_OPAMP_SetMode(
OPAMP_TypeDef * OPAMPx, uint32_t Mode)`**

Function description Set OPAMP mode calibration or functional.

Parameters • **OPAMPx:** OPAMP instance

• **Mode:** This parameter can be one of the following values:

- LL_OPAMP_MODE_FUNCTIONAL
- LL_OPAMP_MODE_CALIBRATION

Return values • **None:**

Notes • OPAMP mode corresponds to functional or calibration mode:
functional mode: OPAMP operation in standalone, follower, ...
Set functional mode using function
`LL_OPAMP_SetFunctionalMode()`.calibration mode: offset
calibration of the selected transistors differential pair NMOS
or PMOS.
• On this STM32 serie, entering in calibration mode makes
loosing OPAMP internal switches configuration. Therefore,
when going back to functional mode, functional mode must be
set again using `LL_OPAMP_SetFunctionalMode()`.

Reference Manual to
LL API cross
reference:

- CSR S3SELx LL_OPAMP_SetMode
- CSR S4SELx LL_OPAMP_SetMode
- CSR S5SELx LL_OPAMP_SetMode
- CSR S6SELx LL_OPAMP_SetMode
- CSR S7SEL2 LL_OPAMP_SetMode

LL_OPAMP_GetMode

Function name	<code>_STATIC_INLINE uint32_t LL_OPAMP_GetMode (OPAMP_TypeDef * OPAMPx)</code>
Function description	Get OPAMP mode calibration or functional.
Parameters	<ul style="list-style-type: none"> • OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_OPAMP_MODE_FUNCTIONAL - LL_OPAMP_MODE_CALIBRATION
Notes	<ul style="list-style-type: none"> • OPAMP mode corresponds to functional or calibration mode: functional mode: OPAMP operation in standalone, follower, ... Set functional mode using function <code>LL_OPAMP_SetFunctionalMode()</code>.calibration mode: offset calibration of the selected transistors differential pair NMOS or PMOS.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR S3SELx <code>LL_OPAMP_SetMode</code> • CSR S4SELx <code>LL_OPAMP_SetMode</code> • CSR S5SELx <code>LL_OPAMP_SetMode</code> • CSR S6SELx <code>LL_OPAMP_SetMode</code> • CSR S7SEL2 <code>LL_OPAMP_SetMode</code>

LL_OPAMP_SetFunctionalMode

Function name	<code>_STATIC_INLINE void LL_OPAMP_SetFunctionalMode (OPAMP_TypeDef * OPAMPx, uint32_t FunctionalMode)</code>
Function description	Set OPAMP functional mode by setting internal connections.
Parameters	<ul style="list-style-type: none"> • OPAMPx: OPAMP instance • FunctionalMode: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_OPAMP_MODE_STANDALONE - LL_OPAMP_MODE_FOLLOWER
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This function reset bit of calibration mode to ensure to be in functional mode, in order to have OPAMP parameters (inputs selection, ...) set with the corresponding OPAMP mode to be effective.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR S3SELx <code>LL_OPAMP_SetFunctionalMode</code>

LL_OPAMP_GetFunctionalMode

Function name	<code>_STATIC_INLINE uint32_t LL_OPAMP_GetFunctionalMode (OPAMP_TypeDef * OPAMPx)</code>
---------------	--

Function description	Get OPAMP functional mode from setting of internal connections.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_OPAMP_MODE_STANDALONE - LL_OPAMP_MODE_FOLLOWER
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR S3SELx LL_OPAMP_GetFunctionalMode

LL_OPAMP_SetInputNonInverting

Function name	<code>__STATIC_INLINE void LL_OPAMP_SetInputNonInverting(OPAMP_TypeDef * OPAMPx, uint32_t InputNonInverting)</code>
Function description	Set OPAMP non-inverting input connection.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance InputNonInverting: This parameter can be one of the following values: (1) Parameter specific to OPAMP instances: OPAMP1, OPAMP2. <ul style="list-style-type: none"> - LL_OPAMP_INPUT_NONINVERT_IO0 - LL_OPAMP_INPUT_NONINV_DAC1_CH1 (1) - LL_OPAMP_INPUT_NONINV_DAC1_CH2 (2) (2) Parameter specific to OPAMP instances: OPAMP2, OPAMP3.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR S5SELx LL_OPAMP_SetInputNonInverting • CSR S6SELx LL_OPAMP_SetInputNonInverting • CSR S7SEL2 LL_OPAMP_SetInputNonInverting

LL_OPAMP_GetInputNonInverting

Function name	<code>__STATIC_INLINE uint32_t LL_OPAMP_GetInputNonInverting(OPAMP_TypeDef * OPAMPx)</code>
Function description	Get OPAMP non-inverting input connection.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: (1) Parameter specific to OPAMP instances: OPAMP1, OPAMP2. <ul style="list-style-type: none"> - LL_OPAMP_INPUT_NONINVERT_IO0 - LL_OPAMP_INPUT_NONINV_DAC1_CH1 (1) - LL_OPAMP_INPUT_NONINV_DAC1_CH2 (2) (2) Parameter specific to OPAMP instances: OPAMP2, OPAMP3.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR S5SELx LL_OPAMP_GetInputNonInverting • CSR S6SELx LL_OPAMP_GetInputNonInverting •

- CSR S7SEL2 LL_OPAMP_SetInputInverting

LL_OPAMP_SetInputInverting

Function name	<code>__STATIC_INLINE void LL_OPAMP_SetInputInverting(OPAMP_TypeDef * OPAMPx, uint32_t InputInverting)</code>
Function description	Set OPAMP inverting input connection.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance InputInverting: This parameter can be one of the following values: (1) Alternative IO pin, not low leakage, availability depends on STM32L1 serie devices packages. <ul style="list-style-type: none"> - LL_OPAMP_INPUT_INVERT_IO0 - LL_OPAMP_INPUT_INVERT_IO1 (1) - LL_OPAMP_INPUT_INVERT_CONNECT_NO
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> OPAMP inverting input is used with OPAMP in mode standalone. Otherwise (OPAMP in mode follower), OPAMP inverting input is not used (not connected to GPIO pin).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR S4SELx LL_OPAMP_SetInputInverting CSR ANAWSELx LL_OPAMP_SetInputInverting

LL_OPAMP_GetInputInverting

Function name	<code>__STATIC_INLINE uint32_t LL_OPAMP_GetInputInverting(OPAMP_TypeDef * OPAMPx)</code>
Function description	Get OPAMP inverting input connection.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: (1) Alternative IO pin, not low leakage, availability depends on STM32L1 serie devices packages. <ul style="list-style-type: none"> - LL_OPAMP_INPUT_INVERT_IO0 - LL_OPAMP_INPUT_INVERT_IO1 (1) - LL_OPAMP_INPUT_INVERT_CONNECT_NO
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR S4SELx LL_OPAMP_SetInputInverting CSR ANAWSELx LL_OPAMP_SetInputInverting

LL_OPAMP_SetCommonTrimmingMode

Function name	<code>__STATIC_INLINE void LL_OPAMP_SetCommonTrimmingMode (OPAMP_Common_TypeDef * OPAMPxy_COMMON, uint32_t TrimmingMode)</code>
Function description	Set OPAMP trimming mode.
Parameters	<ul style="list-style-type: none"> OPAMPxy_COMMON: OPAMP common instance (can be set directly from CMSIS definition or by using helper macro <code>__LL_OPAMP_COMMON_INSTANCE()</code>)

	<ul style="list-style-type: none"> • TrimmingMode: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_OPAMP_TRIMMING_FACTORY – LL_OPAMP_TRIMMING_USER
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • The OPAMP trimming mode applies to several OPAMP instances (if several OPAMP instances available on the selected device).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • OTR OT_USER LL_OPAMP_SetCommonTrimmingMode

LL_OPAMP_GetCommonTrimmingMode

Function name	<code>__STATIC_INLINE uint32_t LL_OPAMP_GetCommonTrimmingMode (OPAMP_Common_TypeDef * OPAMPxy_COMMON)</code>
Function description	Get OPAMP trimming mode.
Parameters	<ul style="list-style-type: none"> • OPAMPxy_COMMON: OPAMP common instance (can be set directly from CMSIS definition or by using helper macro <code>__LL_OPAMP_COMMON_INSTANCE()</code>)
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_OPAMP_TRIMMING_FACTORY – LL_OPAMP_TRIMMING_USER
Notes	<ul style="list-style-type: none"> • The OPAMP trimming mode applies to several OPAMP instances (if several OPAMP instances available on the selected device).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • OTR OT_USER LL_OPAMP_GetCommonTrimmingMode

LL_OPAMP_SetCalibrationSelection

Function name	<code>__STATIC_INLINE void LL_OPAMP_SetCalibrationSelection (OPAMP_TypeDef * OPAMPx, uint32_t TransistorsDiffPair)</code>
Function description	Set OPAMP offset to calibrate the selected transistors differential pair NMOS or PMOS.
Parameters	<ul style="list-style-type: none"> • OPAMPx: OPAMP instance • TransistorsDiffPair: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_OPAMP_TRIMMING_NMOS – LL_OPAMP_TRIMMING_PMOS – LL_OPAMP_TRIMMING_NONE
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Preliminarily, OPAMP must be set in mode calibration using function <code>LL_OPAMP_SetMode()</code>.
Reference Manual to	<ul style="list-style-type: none"> • CSR OPA1CAL_H LL_OPAMP_SetCalibrationSelection

LL API cross reference:	<ul style="list-style-type: none"> CSR OPA1CAL_L LL_OPAMP_SetCalibrationSelection
Function name	<code>__STATIC_INLINE uint32_t LL_OPAMP_GetCalibrationSelection (OPAMP_TypeDef * OPAMPx)</code>
Function description	Get OPAMP offset to calibrate the selected transistors differential pair NMOS or PMOS.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> LL_OPAMP_TRIMMING_NMOS LL_OPAMP_TRIMMING_PMOS LL_OPAMP_TRIMMING_NONE
Notes	<ul style="list-style-type: none"> Preliminarily, OPAMP must be set in mode calibration using function LL_OPAMP_SetMode().
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR OPA1CAL_H LL_OPAMP_SetCalibrationSelection CSR OPA1CAL_L LL_OPAMP_SetCalibrationSelection

LL_OPAMP_IsCalibrationOutputSet

Function name	<code>__STATIC_INLINE uint32_t LL_OPAMP_IsCalibrationOutputSet (OPAMP_TypeDef * OPAMPx)</code>
Function description	Get OPAMP calibration result of toggling output.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> This functions returns: 0 if OPAMP calibration output is reset 1 if OPAMP calibration output is set
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR OPAxCALOUT LL_OPAMP_IsCalibrationOutputSet

LL_OPAMP_SetTrimmingValue

Function name	<code>__STATIC_INLINE void LL_OPAMP_SetTrimmingValue (OPAMP_TypeDef * OPAMPx, uint32_t PowerMode, uint32_t TransistorsDiffPair, uint32_t TrimmingValue)</code>
Function description	Set OPAMP trimming factor for the selected transistors differential pair NMOS or PMOS, corresponding to the selected power mode.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance PowerMode: This parameter can be one of the following values: <ul style="list-style-type: none"> LL_OPAMP_POWERMODE_NORMAL LL_OPAMP_POWERMODE_LOWPOWER TransistorsDiffPair: This parameter can be one of the

	following values: – LL_OPAMP_TRIMMING_NMOS – LL_OPAMP_TRIMMING_PMOS
• TrimmingValue: 0x00...0x1F	
Return values	• None:
Notes	• On STM32L1 serie, OPAMP trimming mode must be re-configured at each update of trimming values in power mode normal. Refer to function LL_OPAMP_SetCommonTrimmingMode().
Reference Manual to LL API cross reference:	• OTR AOx_OPT_OFFSET_TRIM_HIGH LL_OPAMP_SetTrimmingValue • OTR AOx_OPT_OFFSET_TRIM_LOW LL_OPAMP_SetTrimmingValue • LPOTR AOx_OPT_OFFSET_TRIM_LP_HIGH LL_OPAMP_SetTrimmingValue • LPOTR AOx_OPT_OFFSET_TRIM_LP_LOW LL_OPAMP_SetTrimmingValue

LL_OPAMP_GetTrimmingValue

Function name	<code>__STATIC_INLINE uint32_t LL_OPAMP_GetTrimmingValue (OPAMP_TypeDef * OPAMPx, uint32_t PowerMode, uint32_t TransistorsDiffPair)</code>
Function description	Get OPAMP trimming factor for the selected transistors differential pair NMOS or PMOS, corresponding to the selected power mode.
Parameters	<ul style="list-style-type: none"> • OPAMPx: OPAMP instance • PowerMode: This parameter can be one of the following values: – LL_OPAMP_POWERMODE_NORMAL – LL_OPAMP_POWERMODE_LOWPOWER • TransistorsDiffPair: This parameter can be one of the following values: – LL_OPAMP_TRIMMING_NMOS – LL_OPAMP_TRIMMING_PMOS
Return values	• 0x0...0x1F:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • OTR AOx_OPT_OFFSET_TRIM_HIGH LL_OPAMP_GetTrimmingValue • OTR AOx_OPT_OFFSET_TRIM_LOW LL_OPAMP_GetTrimmingValue • LPOTR AOx_OPT_OFFSET_TRIM_LP_HIGH LL_OPAMP_GetTrimmingValue • LPOTR AOx_OPT_OFFSET_TRIM_LP_LOW LL_OPAMP_GetTrimmingValue

LL_OPAMP_Enable

Function name	<code>__STATIC_INLINE void LL_OPAMP_Enable (OPAMP_TypeDef * OPAMPx)</code>
Function description	Enable OPAMP instance.

Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> After enable from off state, OPAMP requires a delay to fulfill wake up time specification. Refer to device datasheet, parameter "tWAKEUP".
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR OPAXPD LL_OPAMP_Enable

LL_OPAMP_Disable

Function name	<code>_STATIC_INLINE void LL_OPAMP_Disable (OPAMP_TypeDef * OPAMPx)</code>
Function description	Disable OPAMP instance.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR OPAXPD LL_OPAMP_Disable

LL_OPAMP_IsEnabled

Function name	<code>_STATIC_INLINE uint32_t LL_OPAMP_IsEnabled (OPAMP_TypeDef * OPAMPx)</code>
Function description	Get OPAMP instance enable state (0: OPAMP is disabled, 1: OPAMP is enabled)
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CSR OPAXPD LL_OPAMP_IsEnabled

LL_OPAMP_DeInit

Function name	<code>ErrorStatus LL_OPAMP_DeInit (OPAMP_TypeDef * OPAMPx)</code>
Function description	De-initialize registers of the selected OPAMP instance to their default reset values.
Parameters	<ul style="list-style-type: none"> OPAMPx: OPAMP instance
Return values	<ul style="list-style-type: none"> An: ErrorStatus enumeration value: <ul style="list-style-type: none"> SUCCESS: OPAMP registers are de-initialized ERROR: OPAMP registers are not de-initialized

LL_OPAMP_Init

Function name	<code>ErrorStatus LL_OPAMP_Init (OPAMP_TypeDef * OPAMPx, LL_OPAMP_InitTypeDef * OPAMP_InitStruct)</code>
---------------	--

Function description	Initialize some features of OPAMP instance.
Parameters	<ul style="list-style-type: none"> • OPAMPx: OPAMP instance • OPAMP_InitStruct: Pointer to a LL_OPAMP_InitTypeDef structure
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: OPAMP registers are initialized – ERROR: OPAMP registers are not initialized
Notes	<ul style="list-style-type: none"> • This function reset bit of calibration mode to ensure to be in functional mode, in order to have OPAMP parameters (inputs selection, ...) set with the corresponding OPAMP mode to be effective. • This function configures features of the selected OPAMP instance. Some features are also available at scope OPAMP common instance (common to several OPAMP instances). Refer to functions having argument "OPAMPxy_COMMON" as parameter.

LL_OPAMP_StructInit

Function name	void LL_OPAMP_StructInit (LL_OPAMP_InitTypeDef * OPAMP_InitStruct)
Function description	Set each LL_OPAMP_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • OPAMP_InitStruct: pointer to a LL_OPAMP_InitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> • None:

60.3 OPAMP Firmware driver defines

60.3.1 OPAMP

OPAMP functional mode

LL_OPAMP_MODE_STANDALONE	OPAMP functional mode, OPAMP operation in standalone (on STM32L1 serie, it corresponds to OPAMP internal switches S3 opened (switch SanB state depends on switch S4 state))
LL_OPAMP_MODE_FOLLOWER	OPAMP functional mode, OPAMP operation in follower (on STM32L1 serie, it corresponds to OPAMP internal switches S3 and SanB closed)

Definitions of OPAMP hardware constraints delays

LL_OPAMP_DELAY_STARTUP_US	Delay for OPAMP startup time
----------------------------------	------------------------------

OPAMP input inverting

LL_OPAMP_INPUT_INVERT_IO0	OPAMP inverting input connected to GPIO pin (low leakage input). Note: OPAMP inverting input is used with OPAMP in mode standalone. Otherwise (OPAMP in mode follower), OPAMP inverting input is not used (not connected)
----------------------------------	---

to GPIO pin).

`LL_OPAMP_INPUT_INVERT_IO1`

OPAMP inverting input connected to GPIO pin (alternative IO pin, not low leakage, availability depends on STM32L1 serie devices packages). Note: OPAMP inverting input is used with OPAMP in mode standalone. Otherwise (OPAMP in mode follower), OPAMP inverting input is not used (not connected to GPIO pin).

`LL_OPAMP_INPUT_INVERT_CONNECT_NO`

OPAMP inverting input not externally connected (intended for OPAMP in mode follower)

OPAMP input non-inverting

`LL_OPAMP_INPUT_NONINVERT_IO0`

OPAMP non inverting input connected to GPIO pin (low leakage input)

`LL_OPAMP_INPUT_NONINV_DAC1_CH1`

OPAMP non inverting input connected to DAC1 channel1 output (specific to OPAMP instances: OPAMP1, OPAMP2)

`LL_OPAMP_INPUT_NONINV_DAC1_CH2`

OPAMP non inverting input connected to DAC1 channel2 output (specific to OPAMP instances: OPAMP2, OPAMP3)

`LL_OPAMP_INPUT_NONINV_DAC1_CH2_OPAMP3`

OPAMP non inverting input connected to DAC1 channel2 output (specific to OPAMP instances: OPAMP3)

OPAMP mode calibration or functional.

`LL_OPAMP_MODE_FUNCTIONAL` OPAMP functional mode

`LL_OPAMP_MODE_CALIBRATION` OPAMP calibration mode (on STM32L1 serie, it corresponds to all OPAMP input internal switches opened)

OPAMP power mode

`LL_OPAMP_POWERMODE_NORMAL` OPAMP power mode normal

`LL_OPAMP_POWERMODE_LOWPOWER` OPAMP power mode low-power

OPAMP power supply range

`LL_OPAMP_POWERSUPPLY_RANGE_LOW` Power supply range low. On STM32L1 serie: Vdda lower than 2.4V.

`LL_OPAMP_POWERSUPPLY_RANGE_HIGH` Power supply range high. On STM32L1 serie: Vdda higher than 2.4V.

OPAMP trimming mode

`LL_OPAMP_TRIMMING_FACTORY` OPAMP trimming factors set to factory values

`LL_OPAMP_TRIMMING_USER` OPAMP trimming factors set to user values

OPAMP trimming of transistors differential pair NMOS or PMOS

LL_OPAMP_TRIMMING_NMOS	OPAMP trimming of transistors differential pair NMOS
LL_OPAMP_TRIMMING_PMOS	OPAMP trimming of transistors differential pair PMOS
LL_OPAMP_TRIMMING_NONE	OPAMP trimming unselect transistors differential pair NMOS and PMOs

OPAMP helper macro_LL_OPAMP_COMMON_INSTANCE**Description:**

- Helper macro to select the OPAMP common instance to which is belonging the selected OPAMP instance.

Parameters:

- __OPAMPx__: OPAMP instance

Return value:

- OPAMP: common instance

Notes:

- OPAMP common register instance can be used to set parameters common to several OPAMP instances. Refer to functions having argument "OPAMPxy_COMMON" as parameter.

_LL_OPAMP_IS_ENABLED_ALL_COMMON_INSTANCE**Description:**

- Helper macro to check if all OPAMP instances sharing the same OPAMP common instance are disabled.

Return value:

- 0: All OPAMP instances sharing the same OPAMP common instance are disabled. 1: At least one OPAMP instance sharing the same OPAMP common instance is enabled

Notes:

- This check is required by functions with setting conditioned to OPAMP state: All OPAMP instances of the OPAMP common group must be disabled. Refer to functions having argument "OPAMPxy_COMMON" as parameter.

Common write and read registers macroLL_OPAMP_WriteReg**Description:**

- Write a value in OPAMP register.

Parameters:

- __INSTANCE__: OPAMP Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_OPAMP_ReadReg

Description:

- Read a value in OPAMP register.

Parameters:

- __INSTANCE__: OPAMP Instance
- __REG__: Register to be read

Return value:

- Register: value

61 LL PWR Generic Driver

61.1 PWR Firmware driver API description

61.1.1 Detailed description of functions

LL_PWR_EnableLowPowerRunMode

Function name **`__STATIC_INLINE void LL_PWR_EnableLowPowerRunMode(void)`**

Function description Switch the Regulator from main mode to low-power mode.

Return values • **None:**

Notes • Remind to set the Regulator to low power before enabling LowPower run mode (bit `LL_PWR_REGU_LPMODES_LOW_POWER`).

Reference Manual to CR LPRUN LL_PWR_EnableLowPowerRunMode
LL API cross reference:

LL_PWR_DisableLowPowerRunMode

Function name **`__STATIC_INLINE void LL_PWR_DisableLowPowerRunMode(void)`**

Function description Switch the Regulator from low-power mode to main mode.

Return values • **None:**

Reference Manual to CR LPRUN LL_PWR_DisableLowPowerRunMode
LL API cross reference:

LL_PWR_IsEnabledLowPowerRunMode

Function name **`__STATIC_INLINE uint32_t LL_PWR_IsEnabledLowPowerRunMode(void)`**

Function description Check if the Regulator is in low-power mode.

Return values • **State:** of bit (1 or 0).

Reference Manual to CR LPRUN LL_PWR_IsEnabledLowPowerRunMode
LL API cross reference:

LL_PWR_EnterLowPowerRunMode

Function name **`__STATIC_INLINE void LL_PWR_EnterLowPowerRunMode(void)`**

Function description Set voltage Regulator to low-power and switch from run main mode to run low-power mode.

tion	
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This "high level" function is introduced to provide functional compatibility with other families. Notice that the two registers have to be written sequentially, so this function is not atomic. To assure atomicity you can call separately the following functions: <code>LL_PWR_SetRegulModeLP(LL_PWR_REGU_LPMODES_LOW_POWER);</code> <code>LL_PWR_EnableLowPowerRunMode();</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR LPSDSR LL_PWR_EnterLowPowerRunMode • CR LPRUN LL_PWR_EnterLowPowerRunMode

LL_PWR_ExitLowPowerRunMode

Function name	<code>__STATIC_INLINE void LL_PWR_ExitLowPowerRunMode (void)</code>
Function description	Set voltage Regulator to main and switch from run main mode to low-power mode.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This "high level" function is introduced to provide functional compatibility with other families. Notice that the two registers have to be written sequentially, so this function is not atomic. To assure atomicity you can call separately the following functions: <code>LL_PWR_DisableLowPowerRunMode();LL_PWR_SetRegulModeLP(LL_PWR_REGU_LPMODES_MAIN);</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR LPSDSR LL_PWR_ExitLowPowerRunMode • CR LPRUN LL_PWR_ExitLowPowerRunMode

LL_PWR_SetRegulVoltageScaling

Function name	<code>__STATIC_INLINE void LL_PWR_SetRegulVoltageScaling (uint32_t VoltageScaling)</code>
Function description	Set the main internal Regulator output voltage.
Parameters	<ul style="list-style-type: none"> • VoltageScaling: This parameter can be one of the following values:

- LL_PWR_REGU_VOLTAGE_SCALE1
- LL_PWR_REGU_VOLTAGE_SCALE2
- LL_PWR_REGU_VOLTAGE_SCALE3

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CR VOS LL_PWR_SetRegulVoltageScaling

LL_PWR_GetRegulVoltageScaling

Function name **`__STATIC_INLINE uint32_t LL_PWR_GetRegulVoltageScaling(void)`**

Function description

Get the main internal Regulator output voltage.

Return values

- **Returned:** value can be one of the following values:
 - LL_PWR_REGU_VOLTAGE_SCALE1
 - LL_PWR_REGU_VOLTAGE_SCALE2
 - LL_PWR_REGU_VOLTAGE_SCALE3

Reference Manual to
LL API cross
reference:

- CR VOS LL_PWR_GetRegulVoltageScaling

LL_PWR_EnableBkUpAccess

Function name **`__STATIC_INLINE void LL_PWR_EnableBkUpAccess(void)`**

Function description

Enable access to the backup domain.

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CR DBP LL_PWR_EnableBkUpAccess

LL_PWR_DisableBkUpAccess

Function name **`__STATIC_INLINE void LL_PWR_DisableBkUpAccess(void)`**

Function description

Disable access to the backup domain.

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CR DBP LL_PWR_DisableBkUpAccess

LL_PWR_IsEnabledBkUpAccess

Function name **`__STATIC_INLINE uint32_t LL_PWR_IsEnabledBkUpAccess(void)`**

Function description

Check if the backup domain is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to
LL API cross

- CR DBP LL_PWR_IsEnabledBkUpAccess

reference:

LL_PWR_SetRegulModeLP

Function name	<code>__STATIC_INLINE void LL_PWR_SetRegulModeLP (uint32_t RegulMode)</code>
Function description	Set voltage Regulator mode during low power modes.
Parameters	<ul style="list-style-type: none"> • RegulMode: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_PWR_REGU_LPMODES_MAIN</code> - <code>LL_PWR_REGU_LPMODES_LOW_POWER</code>
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR LPSDSR LL_PWR_SetRegulModeLP

LL_PWR_GetRegulModeLP

Function name	<code>__STATIC_INLINE uint32_t LL_PWR_GetRegulModeLP (void)</code>
Function description	Get voltage Regulator mode during low power modes.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - <code>LL_PWR_REGU_LPMODES_MAIN</code> - <code>LL_PWR_REGU_LPMODES_LOW_POWER</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR LPSDSR LL_PWR_GetRegulModeLP

LL_PWR_SetPowerMode

Function name	<code>__STATIC_INLINE void LL_PWR_SetPowerMode (uint32_t PDMode)</code>
Function description	Set Power Down mode when CPU enters deepsleep.
Parameters	<ul style="list-style-type: none"> • PDMode: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_PWR_MODE_STOP</code> - <code>LL_PWR_MODE_STANDBY</code>
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Set the Regulator to low power (bit <code>LL_PWR_REGU_LPMODES_LOW_POWER</code>) before setting <code>MODE_STOP</code>. If the Regulator remains in "main mode", it consumes more power without providing any additional feature. In <code>MODE_STANDBY</code> the Regulator is automatically off.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR PDSS LL_PWR_SetPowerMode

LL_PWR_GetPowerMode

Function name	<code>__STATIC_INLINE uint32_t LL_PWR_GetPowerMode (void)</code>
Function description	Get Power Down mode when CPU enters deepsleep.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - <code>LL_PWR_MODE_STOP</code> - <code>LL_PWR_MODE_STANDBY</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR PDDS LL_PWR_GetPowerMode

LL_PWR_SetPVDLevel

Function name	<code>__STATIC_INLINE void LL_PWR_SetPVDLevel (uint32_t PVDLevel)</code>
Function description	Configure the voltage threshold detected by the Power Voltage Detector.
Parameters	<ul style="list-style-type: none"> • PVDLevel: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_PWR_PVDLEVEL_0</code> - <code>LL_PWR_PVDLEVEL_1</code> - <code>LL_PWR_PVDLEVEL_2</code> - <code>LL_PWR_PVDLEVEL_3</code> - <code>LL_PWR_PVDLEVEL_4</code> - <code>LL_PWR_PVDLEVEL_5</code> - <code>LL_PWR_PVDLEVEL_6</code> - <code>LL_PWR_PVDLEVEL_7</code>
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR PLS LL_PWR_SetPVDLevel

LL_PWR_GetPVDLevel

Function name	<code>__STATIC_INLINE uint32_t LL_PWR_GetPVDLevel (void)</code>
Function description	Get the voltage threshold detection.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - <code>LL_PWR_PVDLEVEL_0</code> - <code>LL_PWR_PVDLEVEL_1</code> - <code>LL_PWR_PVDLEVEL_2</code> - <code>LL_PWR_PVDLEVEL_3</code> - <code>LL_PWR_PVDLEVEL_4</code> - <code>LL_PWR_PVDLEVEL_5</code> - <code>LL_PWR_PVDLEVEL_6</code> - <code>LL_PWR_PVDLEVEL_7</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR PLS LL_PWR_GetPVDLevel

LL_PWR_EnablePVD

Function name	<code>__STATIC_INLINE void LL_PWR_EnablePVD (void)</code>
Function description	Enable Power Voltage Detector.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR PVDE LL_PWR_EnablePVD

LL_PWR_DisablePVD

Function name	<code>__STATIC_INLINE void LL_PWR_DisablePVD (void)</code>
Function description	Disable Power Voltage Detector.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR PVDE LL_PWR_DisablePVD

LL_PWR_IsEnabledPVD

Function name	<code>__STATIC_INLINE uint32_t LL_PWR_IsEnabledPVD (void)</code>
Function description	Check if Power Voltage Detector is enabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR PVDE LL_PWR_IsEnabledPVD

LL_PWR_EnableWakeUpPin

Function name	<code>__STATIC_INLINE void LL_PWR_EnableWakeUpPin (uint32_t WakeUpPin)</code>
Function description	Enable the WakeUp PINx functionality.
Parameters	<ul style="list-style-type: none"> • WakeUpPin: This parameter can be one of the following values: (*) not available on all devices <ul style="list-style-type: none"> - LL_PWR_WAKEUP_PIN1 - LL_PWR_WAKEUP_PIN2 - LL_PWR_WAKEUP_PIN3 (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR EWUP1 LL_PWR_EnableWakeUpPin • CSR EWUP2 LL_PWR_EnableWakeUpPin • CSR EWUP3 LL_PWR_EnableWakeUpPin

LL_PWR_DisableWakeUpPin

Function name	<code>__STATIC_INLINE void LL_PWR_DisableWakeUpPin (uint32_t</code>
---------------	---

WakeUpPin)

Function description	Disable the WakeUp PINx functionality.
Parameters	<ul style="list-style-type: none"> • WakeUpPin: This parameter can be one of the following values: (*) not available on all devices <ul style="list-style-type: none"> – LL_PWR_WAKEUP_PIN1 – LL_PWR_WAKEUP_PIN2 – LL_PWR_WAKEUP_PIN3 (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR EWUP1 LL_PWR_DisableWakeUpPin • CSR EWUP2 LL_PWR_DisableWakeUpPin • CSR EWUP3 LL_PWR_DisableWakeUpPin

LL_PWR_IsEnabledWakeUpPin

Function name	_STATIC_INLINE uint32_t LL_PWR_IsEnabledWakeUpPin (uint32_t WakeUpPin)
Function description	Check if the WakeUp PINx functionality is enabled.
Parameters	<ul style="list-style-type: none"> • WakeUpPin: This parameter can be one of the following values: (*) not available on all devices <ul style="list-style-type: none"> – LL_PWR_WAKEUP_PIN1 – LL_PWR_WAKEUP_PIN2 – LL_PWR_WAKEUP_PIN3 (*)
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR EWUP1 LL_PWR_IsEnabledWakeUpPin • CSR EWUP2 LL_PWR_IsEnabledWakeUpPin • CSR EWUP3 LL_PWR_IsEnabledWakeUpPin

LL_PWR_EnableUltraLowPower

Function name	_STATIC_INLINE void LL_PWR_EnableUltraLowPower (void)
Function description	Enable ultra low-power mode by enabling VREFINT switch off in low-power modes.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR ULP LL_PWR_EnableUltraLowPower

LL_PWR_DisableUltraLowPower

Function name	_STATIC_INLINE void LL_PWR_DisableUltraLowPower (void)
Function description	Disable ultra low-power mode by disabling VREFINT switch off in

low-power modes.

Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR ULP LL_PWR_DisableUltraLowPower

LL_PWR_IsEnabledUltraLowPower

Function name	<code>__STATIC_INLINE uint32_t LL_PWR_IsEnabledUltraLowPower(void)</code>
Function description	Check if ultra low-power mode is enabled by checking if VREFINT switch off in low-power modes is enabled.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR ULP LL_PWR_IsEnabledUltraLowPower

LL_PWR_EnableFastWakeUp

Function name	<code>__STATIC_INLINE void LL_PWR_EnableFastWakeUp(void)</code>
Function description	Enable fast wakeup by ignoring VREFINT startup time when exiting from low-power mode.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Works in conjunction with ultra low power mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR FWU LL_PWR_EnableFastWakeUp

LL_PWR_DisableFastWakeUp

Function name	<code>__STATIC_INLINE void LL_PWR_DisableFastWakeUp(void)</code>
Function description	Disable fast wakeup by waiting VREFINT startup time when exiting from low-power mode.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Works in conjunction with ultra low power mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR FWU LL_PWR_DisableFastWakeUp

LL_PWR_IsEnabledFastWakeUp

Function name	<code>__STATIC_INLINE uint32_t LL_PWR_IsEnabledFastWakeUp(void)</code>
Function description	Check if fast wakeup is enabled by checking if VREFINT startup time when exiting from low-power mode is ignored.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).

Reference Manual to
LL API cross
reference:

- CR FWU LL_PWR_IsEnabledFastWakeUp

LL_PWR_IsActiveFlag_WU

Function name **`__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_WU (void)`**

Function description Get Wake-up Flag.

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
CSR WUF LL_PWR_IsActiveFlag_WU

LL_PWR_IsActiveFlag_SB

Function name **`__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_SB (void)`**

Function description Get Standby Flag.

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
CSR SBF LL_PWR_IsActiveFlag_SB

LL_PWR_IsActiveFlag_PVDO

Function name **`__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_PVDO (void)`**

Function description Indicate whether VDD voltage is below the selected PVD threshold.

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
CSR PVDO LL_PWR_IsActiveFlag_PVDO

LL_PWR_IsActiveFlag_VREFINTRDY

Function name **`__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_VREFINTRDY (void)`**

Function description Get Internal Reference VrefInt Flag.

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
CSR VREFINTRDYF LL_PWR_IsActiveFlag_VREFINTRDY

LL_PWR_IsActiveFlag_VOS

Function name **`__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_VOS (void)`**

Function description Indicate whether the Regulator is ready in the selected voltage range or if its output voltage is still changing to the required

voltage level.

Return values

- **State:** of bit (1 or 0).

**Reference Manual to
LL API cross
reference:**

- CSR VOSF LL_PWR_IsActiveFlag_VOS

LL_PWR_IsActiveFlag_REGLPF

Function name

**`__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_REGLPF(
(void)`**

Function description

Indicate whether the Regulator is ready in main mode or is in low-power mode.

Return values

- **State:** of bit (1 or 0).

Notes

- Take care, return value "0" means the Regulator is ready. Return value "1" means the output voltage range is still changing.

**Reference Manual to
LL API cross
reference:**

- CSR REGLPF LL_PWR_IsActiveFlag_REGLPF

LL_PWR_ClearFlag_SB

Function name

`__STATIC_INLINE void LL_PWR_ClearFlag_SB (void)`

Function description

Clear Standby Flag.

Return values

- **None:**

**Reference Manual to
LL API cross
reference:**

- CR CSBF LL_PWR_ClearFlag_SB

LL_PWR_ClearFlag_WU

Function name

`__STATIC_INLINE void LL_PWR_ClearFlag_WU (void)`

Function description

Clear Wake-up Flags.

Return values

- **None:**

**Reference Manual to
LL API cross
reference:**

- CR CWUF LL_PWR_ClearFlag_WU

LL_PWR_DeInit

Function name

`ErrorStatus LL_PWR_DeInit (void)`

Function description

De-initialize the PWR registers to their default reset values.

Return values

- **An:** ErrorStatus enumeration value:
 - SUCCESS: PWR registers are de-initialized
 - ERROR: not applicable

61.2 PWR Firmware driver defines

61.2.1 PWR

Clear Flags Defines

`LL_PWR_CR_CSBF` Clear standby flag

`LL_PWR_CR_CWUF` Clear wakeup flag

Get Flags Defines

`LL_PWR_CSR_WUF` Wakeup flag

`LL_PWR_CSR_SBF` Standby flag

`LL_PWR_CSR_PVDO` Power voltage detector output flag

`LL_PWR_CSR_VREFINTRDYF` VREFINT ready flag

`LL_PWR_CSR_VOS` Voltage scaling select flag

`LL_PWR_CSR_REGLPF` Regulator low power flag

`LL_PWR_CSR_EWUP1` Enable WKUP pin 1

`LL_PWR_CSR_EWUP2` Enable WKUP pin 2

`LL_PWR_CSR_EWUP3` Enable WKUP pin 3

Mode Power

`LL_PWR_MODE_STOP` Enter Stop mode when the CPU enters deepsleep

`LL_PWR_MODE_STANDBY` Enter Standby mode when the CPU enters deepsleep

Power Voltage Detector Level

`LL_PWR_PVDLEVEL_0` Voltage threshold detected by PVD 1.9 V

`LL_PWR_PVDLEVEL_1` Voltage threshold detected by PVD 2.1 V

`LL_PWR_PVDLEVEL_2` Voltage threshold detected by PVD 2.3 V

`LL_PWR_PVDLEVEL_3` Voltage threshold detected by PVD 2.5 V

`LL_PWR_PVDLEVEL_4` Voltage threshold detected by PVD 2.7 V

`LL_PWR_PVDLEVEL_5` Voltage threshold detected by PVD 2.9 V

`LL_PWR_PVDLEVEL_6` Voltage threshold detected by PVD 3.1 V

`LL_PWR_PVDLEVEL_7` External input analog voltage (Compare internally to VREFINT)

Regulator Mode In Low Power Modes

`LL_PWR_REGU_LPMODES_MAIN` Voltage Regulator in main mode during deepsleep/sleep/low-power run mode

`LL_PWR_REGU_LPMODES_LOW_POWER` Voltage Regulator in low-power mode during deepsleep/sleep/low-power run mode

Regulator Voltage

`LL_PWR_REGU_VOLTAGE_SCALE1` 1.8V (range 1)

`LL_PWR_REGU_VOLTAGE_SCALE2` 1.5V (range 2)

`LL_PWR_REGU_VOLTAGE_SCALE3` 1.2V (range 3)

Wakeup Pins

LL_PWR_WAKEUP_PIN1 WKUP pin 1 : PA0

LL_PWR_WAKEUP_PIN2 WKUP pin 2 : PC13

LL_PWR_WAKEUP_PIN3 WKUP pin 3 : PE6 or PA2 according to device

PWR legacy functions name

LL_PWR_IsActiveFlag_VOSF

Common write and read registers Macros

LL_PWR_WriteReg **Description:**

- Write a value in PWR register.

Parameters:

- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_PWR_ReadReg

Description:

- Read a value in PWR register.

Parameters:

- __REG__: Register to be read

Return value:

- Register: value

62 LL RCC Generic Driver

62.1 RCC Firmware driver registers structures

62.1.1 LL_RCC_ClocksTypeDef

Data Fields

- *uint32_t SYSCLK_Frequency*
- *uint32_t HCLK_Frequency*
- *uint32_t PCLK1_Frequency*
- *uint32_t PCLK2_Frequency*

Field Documentation

- *uint32_t LL_RCC_ClocksTypeDef::SYSCLK_Frequency*
SYSCLK clock frequency
- *uint32_t LL_RCC_ClocksTypeDef::HCLK_Frequency*
HCLK clock frequency
- *uint32_t LL_RCC_ClocksTypeDef::PCLK1_Frequency*
PCLK1 clock frequency
- *uint32_t LL_RCC_ClocksTypeDef::PCLK2_Frequency*
PCLK2 clock frequency

62.2 RCC Firmware driver API description

62.2.1 Detailed description of functions

LL_RCC_HSE_EnableCSS

Function name	<code>__STATIC_INLINE void LL_RCC_HSE_EnableCSS (void)</code>
Function description	Enable the Clock Security System.
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR CSSON LL_RCC_HSE_EnableCSS

LL_RCC_HSE_DisableCSS

Function name	<code>__STATIC_INLINE void LL_RCC_HSE_DisableCSS (void)</code>
Function description	Disable the Clock Security System.
Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• Cannot be disabled in HSE is ready (only by hardware)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR CSSON LL_RCC_HSE_DisableCSS

LL_RCC_HSE_EnableBypass

Function name	<code>__STATIC_INLINE void LL_RCC_HSE_EnableBypass (void)</code>
Function description	Enable HSE external oscillator (HSE Bypass)
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR HSEBYP LL_RCC_HSE_EnableBypass

LL_RCC_HSE_DisableBypass

Function name	<code>__STATIC_INLINE void LL_RCC_HSE_DisableBypass (void)</code>
Function description	Disable HSE external oscillator (HSE Bypass)
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR HSEBYP LL_RCC_HSE_DisableBypass

LL_RCC_HSE_Enable

Function name	<code>__STATIC_INLINE void LL_RCC_HSE_Enable (void)</code>
Function description	Enable HSE crystal oscillator (HSE ON)
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR HSEON LL_RCC_HSE_Enable

LL_RCC_HSE_Disable

Function name	<code>__STATIC_INLINE void LL_RCC_HSE_Disable (void)</code>
Function description	Disable HSE crystal oscillator (HSE ON)
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR HSEON LL_RCC_HSE_Disable

LL_RCC_HSE_IsReady

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_HSE_IsReady (void)</code>
Function description	Check if HSE oscillator Ready.
Return values	<ul style="list-style-type: none">• State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR HSERDY LL_RCC_HSE_IsReady

LL_RCC_SetRTC_HSEPrescaler

Function name	<code>__STATIC_INLINE void LL_RCC_SetRTC_HSEPrescaler (uint32_t Div)</code>
Function description	Configure the RTC prescaler (divider)
Parameters	<ul style="list-style-type: none"> • Div: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_RCC_RTC_HSE_DIV_2</code> - <code>LL_RCC_RTC_HSE_DIV_4</code> - <code>LL_RCC_RTC_HSE_DIV_8</code> - <code>LL_RCC_RTC_HSE_DIV_16</code>
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR RTCPRE LL_RCC_SetRTC_HSEPrescaler

LL_RCC_GetRTC_HSEPrescaler

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_GetRTC_HSEPrescaler (void)</code>
Function description	Get the RTC divider (prescaler)
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - <code>LL_RCC_RTC_HSE_DIV_2</code> - <code>LL_RCC_RTC_HSE_DIV_4</code> - <code>LL_RCC_RTC_HSE_DIV_8</code> - <code>LL_RCC_RTC_HSE_DIV_16</code>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR RTCPRE LL_RCC_GetRTC_HSEPrescaler

LL_RCC_HSI_Enable

Function name	<code>__STATIC_INLINE void LL_RCC_HSI_Enable (void)</code>
Function description	Enable HSI oscillator.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR HSION LL_RCC_HSI_Enable

LL_RCC_HSI_Disable

Function name	<code>__STATIC_INLINE void LL_RCC_HSI_Disable (void)</code>
Function description	Disable HSI oscillator.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR HSION LL_RCC_HSI_Disable

LL_RCC_HSI_IsReady

Function name **`_STATIC_INLINE uint32_t LL_RCC_HSI_IsReady (void)`**

Function description Check if HSI clock is ready.

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:

- CR HSIRDY LL_RCC_HSI_IsReady

LL_RCC_HSI_GetCalibration

Function name **`_STATIC_INLINE uint32_t LL_RCC_HSI_GetCalibration (void)`**

Function description Get HSI Calibration value.

Return values • **Between:** Min_Data = 0x00 and Max_Data = 0xFF

Notes • When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value

Reference Manual to
LL API cross
reference:

- ICSCR HSICAL LL_RCC_HSI_GetCalibration

LL_RCC_HSI_SetCalibTrimming

Function name **`_STATIC_INLINE void LL_RCC_HSI_SetCalibTrimming (uint32_t Value)`**

Function description Set HSI Calibration trimming.

Parameters • **Value:** between Min_Data = 0x00 and Max_Data = 0x1F

Return values • **None:**

Notes • user-programmable trimming value that is added to the HSICAL
 • Default value is 16, which, when added to the HSICAL value,
 should trim the HSI to 16 MHz +/- 1 %

Reference Manual to
LL API cross
reference:

- ICSCR HSITRIM LL_RCC_HSI_SetCalibTrimming

LL_RCC_HSI_GetCalibTrimming

Function name **`_STATIC_INLINE uint32_t LL_RCC_HSI_GetCalibTrimming (void)`**

Function description Get HSI Calibration trimming.

Return values • **Between:** Min_Data = 0x00 and Max_Data = 0x1F

Reference Manual to
LL API cross
reference:

- ICSCR HSITRIM LL_RCC_HSI_GetCalibTrimming

LL_RCC_LSE_Enable

Function name	<code>__STATIC_INLINE void LL_RCC_LSE_Enable (void)</code>
Function description	Enable Low Speed External (LSE) crystal.
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CSR LSEON LL_RCC_LSE_Enable

LL_RCC_LSE_Disable

Function name	<code>__STATIC_INLINE void LL_RCC_LSE_Disable (void)</code>
Function description	Disable Low Speed External (LSE) crystal.
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CSR LSEON LL_RCC_LSE_Disable

LL_RCC_LSE_EnableBypass

Function name	<code>__STATIC_INLINE void LL_RCC_LSE_EnableBypass (void)</code>
Function description	Enable external clock source (LSE bypass).
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CSR LSEBYP LL_RCC_LSE_EnableBypass

LL_RCC_LSE_DisableBypass

Function name	<code>__STATIC_INLINE void LL_RCC_LSE_DisableBypass (void)</code>
Function description	Disable external clock source (LSE bypass).
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CSR LSEBYP LL_RCC_LSE_DisableBypass

LL_RCC_LSE_EnableCSS

Function name	<code>__STATIC_INLINE void LL_RCC_LSE_EnableCSS (void)</code>
Function description	Enable Clock security system on LSE.
Return values	<ul style="list-style-type: none">• None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CSR LSECSSON LL_RCC_LSE_EnableCSS

LL_RCC_LSE_DisableCSS

Function name	<code>__STATIC_INLINE void LL_RCC_LSE_DisableCSS (void)</code>
Function description	Disable Clock security system on LSE.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Clock security system can be disabled only after a LSE failure detection. In that case it MUST be disabled by software.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR LSECSSON LL_RCC_LSE_DisableCSS

LL_RCC_LSE_IsReady

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_LSE_IsReady (void)</code>
Function description	Check if LSE oscillator Ready.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR LSERDY LL_RCC_LSE_IsReady

LL_RCC_LSE_IsCSSDetected

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_LSE_IsCSSDetected (void)</code>
Function description	Check if CSS on LSE failure Detection.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR LSECSSD LL_RCC_LSE_IsCSSDetected

LL_RCC_LSI_Enable

Function name	<code>__STATIC_INLINE void LL_RCC_LSI_Enable (void)</code>
Function description	Enable LSI Oscillator.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR LSION LL_RCC_LSI_Enable

LL_RCC_LSI_Disable

Function name	<code>__STATIC_INLINE void LL_RCC_LSI_Disable (void)</code>
Function description	Disable LSI Oscillator.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross	<ul style="list-style-type: none"> • CSR LSION LL_RCC_LSI_Disable

reference:

LL_RCC_LSI_IsReady

Function name **`__STATIC_INLINE uint32_t LL_RCC_LSI_IsReady (void)`**

Function description Check if LSI is Ready.

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
CSR LSIRDY LL_RCC_LSI_IsReady

LL_RCC_MSI_Enable

Function name **`__STATIC_INLINE void LL_RCC_MSI_Enable (void)`**

Function description Enable MSI oscillator.

Return values • **None:**

Reference Manual to
LL API cross
reference:
CR MSION LL_RCC_MSI_Enable

LL_RCC_MSI_Disable

Function name **`__STATIC_INLINE void LL_RCC_MSI_Disable (void)`**

Function description Disable MSI oscillator.

Return values • **None:**

Reference Manual to
LL API cross
reference:
CR MSION LL_RCC_MSI_Disable

LL_RCC_MSI_IsReady

Function name **`__STATIC_INLINE uint32_t LL_RCC_MSI_IsReady (void)`**

Function description Check if MSI oscillator Ready.

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
CR MSIRDY LL_RCC_MSI_IsReady

LL_RCC_MSI_SetRange

Function name **`__STATIC_INLINE void LL_RCC_MSI_SetRange (uint32_t Range)`**

Function description Configure the Internal Multi Speed oscillator (MSI) clock range in run mode.

Parameters • **Range:** This parameter can be one of the following values:
– `LL_RCC_MSIRANGE_0`
– `LL_RCC_MSIRANGE_1`

	<ul style="list-style-type: none"> - LL_RCC_MSIRANGE_2 - LL_RCC_MSIRANGE_3 - LL_RCC_MSIRANGE_4 - LL_RCC_MSIRANGE_5 - LL_RCC_MSIRANGE_6
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ICSCR MSIRANGE LL_RCC_MSI_SetRange

LL_RCC_MSI_GetRange

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_MSI_GetRange (void)</code>
Function description	Get the Internal Multi Speed oscillator (MSI) clock range in run mode.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_MSIRANGE_0 - LL_RCC_MSIRANGE_1 - LL_RCC_MSIRANGE_2 - LL_RCC_MSIRANGE_3 - LL_RCC_MSIRANGE_4 - LL_RCC_MSIRANGE_5 - LL_RCC_MSIRANGE_6
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ICSCR MSIRANGE LL_RCC_MSI_GetRange

LL_RCC_MSI_GetCalibration

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_MSI_GetCalibration (void)</code>
Function description	Get MSI Calibration value.
Return values	<ul style="list-style-type: none"> • Between: Min_Data = 0x00 and Max_Data = 0xFF
Notes	<ul style="list-style-type: none"> • When MSITRIM is written, MSICAL is updated with the sum of MSITRIM and the factory trim value
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ICSCR MSICAL LL_RCC_MSI_GetCalibration

LL_RCC_MSI_SetCalibTrimming

Function name	<code>__STATIC_INLINE void LL_RCC_MSI_SetCalibTrimming (uint32_t Value)</code>
Function description	Set MSI Calibration trimming.
Parameters	<ul style="list-style-type: none"> • Value: between Min_Data = 0x00 and Max_Data = 0xFF
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • user-programmable trimming value that is added to the

MSICAL

Reference Manual to
LL API cross
reference:

- ICSCR MSITRIM LL_RCC_MSI_SetCalibTrimming

LL_RCC_MSI_SetCalibTrimming

Function name **`__STATIC_INLINE uint32_t LL_RCC_MSI_SetCalibTrimming(void)`**

Function description Get MSI Calibration trimming.

Return values

- **Between:** Min_Data = 0x00 and Max_Data = 0xFF

Reference Manual to
LL API cross
reference:

- ICSCR MSITRIM LL_RCC_MSI_SetCalibTrimming

LL_RCC_SetSysClkSource

Function name **`__STATIC_INLINE void LL_RCC_SetSysClkSource (uint32_t Source)`**

Function description Configure the system clock source.

Parameters

- **Source:** This parameter can be one of the following values:
 - LL_RCC_SYS_CLKSOURCE_MSI
 - LL_RCC_SYS_CLKSOURCE_HSI
 - LL_RCC_SYS_CLKSOURCE_HSE
 - LL_RCC_SYS_CLKSOURCE_PLL

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- CFGR SW LL_RCC_SetSysClkSource

LL_RCC_GetSysClkSource

Function name **`__STATIC_INLINE uint32_t LL_RCC_GetSysClkSource (void)`**

Function description Get the system clock source.

Return values

- **Returned:** value can be one of the following values:
 - LL_RCC_SYS_CLKSOURCE_STATUS_MSI
 - LL_RCC_SYS_CLKSOURCE_STATUS_HSI
 - LL_RCC_SYS_CLKSOURCE_STATUS_HSE
 - LL_RCC_SYS_CLKSOURCE_STATUS_PLL

Reference Manual to
LL API cross
reference:

- CFGR SWS LL_RCC_GetSysClkSource

LL_RCC_SetAHBPrescaler

Function name **`__STATIC_INLINE void LL_RCC_SetAHBPrescaler (uint32_t Prescaler)`**

Function description	Set AHB prescaler.
Parameters	<ul style="list-style-type: none"> Prescaler: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_SYSCLK_DIV_1 - LL_RCC_SYSCLK_DIV_2 - LL_RCC_SYSCLK_DIV_4 - LL_RCC_SYSCLK_DIV_8 - LL_RCC_SYSCLK_DIV_16 - LL_RCC_SYSCLK_DIV_64 - LL_RCC_SYSCLK_DIV_128 - LL_RCC_SYSCLK_DIV_256 - LL_RCC_SYSCLK_DIV_512
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CFGR HPRE LL_RCC_SetAHBPrescaler

LL_RCC_SetAPB1Prescaler

Function name	<code>__STATIC_INLINE void LL_RCC_SetAPB1Prescaler (uint32_t Prescaler)</code>
Function description	Set APB1 prescaler.
Parameters	<ul style="list-style-type: none"> Prescaler: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_APB1_DIV_1 - LL_RCC_APB1_DIV_2 - LL_RCC_APB1_DIV_4 - LL_RCC_APB1_DIV_8 - LL_RCC_APB1_DIV_16
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CFGR PPREG1 LL_RCC_SetAPB1Prescaler

LL_RCC_SetAPB2Prescaler

Function name	<code>__STATIC_INLINE void LL_RCC_SetAPB2Prescaler (uint32_t Prescaler)</code>
Function description	Set APB2 prescaler.
Parameters	<ul style="list-style-type: none"> Prescaler: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_APB2_DIV_1 - LL_RCC_APB2_DIV_2 - LL_RCC_APB2_DIV_4 - LL_RCC_APB2_DIV_8 - LL_RCC_APB2_DIV_16
Return values	<ul style="list-style-type: none"> • None:

Reference Manual to
LL API cross
reference:

- CFGR_PPREG2 LL_RCC_SetAPB2Prescaler

LL_RCC_GetAHBPrescaler

Function name **`__STATIC_INLINE uint32_t LL_RCC_GetAHBPrescaler (void)`**

Function description Get AHB prescaler.

Return values

- **Returned:** value can be one of the following values:
 - LL_RCC_SYSCLK_DIV_1
 - LL_RCC_SYSCLK_DIV_2
 - LL_RCC_SYSCLK_DIV_4
 - LL_RCC_SYSCLK_DIV_8
 - LL_RCC_SYSCLK_DIV_16
 - LL_RCC_SYSCLK_DIV_64
 - LL_RCC_SYSCLK_DIV_128
 - LL_RCC_SYSCLK_DIV_256
 - LL_RCC_SYSCLK_DIV_512

Reference Manual to
LL API cross
reference:

- CFGR_HPRE LL_RCC_GetAHBPrescaler

LL_RCC_GetAPB1Prescaler

Function name **`__STATIC_INLINE uint32_t LL_RCC_GetAPB1Prescaler (void)`**

Function description Get APB1 prescaler.

Return values

- **Returned:** value can be one of the following values:
 - LL_RCC_APB1_DIV_1
 - LL_RCC_APB1_DIV_2
 - LL_RCC_APB1_DIV_4
 - LL_RCC_APB1_DIV_8
 - LL_RCC_APB1_DIV_16

Reference Manual to
LL API cross
reference:

- CFGR_PPREG1 LL_RCC_GetAPB1Prescaler

LL_RCC_GetAPB2Prescaler

Function name **`__STATIC_INLINE uint32_t LL_RCC_GetAPB2Prescaler (void)`**

Function description Get APB2 prescaler.

Return values

- **Returned:** value can be one of the following values:
 - LL_RCC_APB2_DIV_1
 - LL_RCC_APB2_DIV_2
 - LL_RCC_APB2_DIV_4
 - LL_RCC_APB2_DIV_8
 - LL_RCC_APB2_DIV_16

Reference Manual to
LL API cross

- CFGR_PPREG2 LL_RCC_GetAPB2Prescaler

reference:

LL_RCC_ConfigMCO

Function name	<code>__STATIC_INLINE void LL_RCC_ConfigMCO (uint32_t MC0xSource, uint32_t MC0xPrescaler)</code>
Function description	Configure MC0x.
Parameters	<ul style="list-style-type: none"> • MC0xSource: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RCC_MCO1SOURCE_NOCLOCK – LL_RCC_MCO1SOURCE_SYSCLK – LL_RCC_MCO1SOURCE_HSI – LL_RCC_MCO1SOURCE_MSI – LL_RCC_MCO1SOURCE_HSE – LL_RCC_MCO1SOURCE_PLLCLK – LL_RCC_MCO1SOURCE_LSI – LL_RCC_MCO1SOURCE_LSE • MC0xPrescaler: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RCC_MCO1_DIV_1 – LL_RCC_MCO1_DIV_2 – LL_RCC_MCO1_DIV_4 – LL_RCC_MCO1_DIV_8 – LL_RCC_MCO1_DIV_16
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CFGR MC0SEL LL_RCC_ConfigMCO • CFGR MCOPRE LL_RCC_ConfigMCO

LL_RCC_SetRTCClockSource

Function name	<code>__STATIC_INLINE void LL_RCC_SetRTCClockSource (uint32_t Source)</code>
Function description	Set RTC Clock Source.
Parameters	<ul style="list-style-type: none"> • Source: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RCC_RTC_CLKSOURCE_NONE – LL_RCC_RTC_CLKSOURCE_LSE – LL_RCC_RTC_CLKSOURCE_LSI – LL_RCC_RTC_CLKSOURCE_HSE
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Once the RTC clock source has been selected, it cannot be changed any more unless the Backup domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The RTCRST bit can be used to reset them.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CSR RTCSEL LL_RCC_SetRTCClockSource

LL_RCC_GetRTCClockSource

Function name **`__STATIC_INLINE uint32_t LL_RCC_GetRTCClockSource(void)`**

Function description Get RTC Clock Source.

Return values

- **Returned:** value can be one of the following values:
 - `LL_RCC_RTC_CLKSOURCE_NONE`
 - `LL_RCC_RTC_CLKSOURCE_LSE`
 - `LL_RCC_RTC_CLKSOURCE_LSI`
 - `LL_RCC_RTC_CLKSOURCE_HSE`

Reference Manual to
LL API cross
reference:
 • CSR RTCSEL LL_RCC_GetRTCClockSource

LL_RCC_EnableRTC

Function name **`__STATIC_INLINE void LL_RCC_EnableRTC(void)`**

Function description Enable RTC.

Return values

- **None:**

Reference Manual to
LL API cross
reference:
 • CSR RTCEN LL_RCC_EnableRTC

LL_RCC_DisableRTC

Function name **`__STATIC_INLINE void LL_RCC_DisableRTC(void)`**

Function description Disable RTC.

Return values

- **None:**

Reference Manual to
LL API cross
reference:
 • CSR RTCEN LL_RCC_DisableRTC

LL_RCC_IsEnabledRTC

Function name **`__STATIC_INLINE uint32_t LL_RCC_IsEnabledRTC(void)`**

Function description Check if RTC has been enabled or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
 • CSR RTCEN LL_RCC_IsEnabledRTC

LL_RCC_ForceBackupDomainReset

Function name **`__STATIC_INLINE void LL_RCC_ForceBackupDomainReset(void)`**

Function description Force the Backup domain reset.

Return values

- **None:**

- Reference Manual to CSR RTCRST LL_RCC_ForceBackupDomainReset
LL API cross reference:
- CSR RTCRST LL_RCC_ForceBackupDomainReset

LL_RCC_ReleaseBackupDomainReset

- Function name **_STATIC_INLINE void LL_RCC_ReleaseBackupDomainReset (void)**
- Function description Release the Backup domain reset.
- Return values • **None:**
- Reference Manual to CSR RTCRST LL_RCC_ReleaseBackupDomainReset
LL API cross reference:
- CSR RTCRST LL_RCC_ReleaseBackupDomainReset

LL_RCC_PLL_Enable

- Function name **_STATIC_INLINE void LL_RCC_PLL_Enable (void)**
- Function description Enable PLL.
- Return values • **None:**
- Reference Manual to CR PLLON LL_RCC_PLL_Enable
LL API cross reference:
- CR PLLON LL_RCC_PLL_Enable

LL_RCC_PLL_Disable

- Function name **_STATIC_INLINE void LL_RCC_PLL_Disable (void)**
- Function description Disable PLL.
- Return values • **None:**
- Notes • Cannot be disabled if the PLL clock is used as the system clock
- Reference Manual to CR PLLON LL_RCC_PLL_Disable
LL API cross reference:
- CR PLLON LL_RCC_PLL_Disable

LL_RCC_PLL_IsReady

- Function name **_STATIC_INLINE uint32_t LL_RCC_PLL_IsReady (void)**
- Function description Check if PLL Ready.
- Return values • **State:** of bit (1 or 0).
- Reference Manual to CR PLLRDY LL_RCC_PLL_IsReady
LL API cross reference:
- CR PLLRDY LL_RCC_PLL_IsReady

LL_RCC_PLL_ConfigDomain_SYS

- Function name **_STATIC_INLINE void LL_RCC_PLL_ConfigDomain_SYS (uint32_t Source, uint32_t PLLMul, uint32_t PLLDiv)**

Function description	Configure PLL used for SYSCLK Domain.
Parameters	<ul style="list-style-type: none"> • Source: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_PLLSOURCE_HSI - LL_RCC_PLLSOURCE_HSE • PLLMul: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_PLL_MUL_3 - LL_RCC_PLL_MUL_4 - LL_RCC_PLL_MUL_6 - LL_RCC_PLL_MUL_8 - LL_RCC_PLL_MUL_12 - LL_RCC_PLL_MUL_16 - LL_RCC_PLL_MUL_24 - LL_RCC_PLL_MUL_32 - LL_RCC_PLL_MUL_48 • PLLDIV: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_PLL_DIV_2 - LL_RCC_PLL_DIV_3 - LL_RCC_PLL_DIV_4
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CFGR PLLSRC LL_RCC_PLL_ConfigDomain_SYS • CFGR PLLMUL LL_RCC_PLL_ConfigDomain_SYS • CFGR PLLDIV LL_RCC_PLL_ConfigDomain_SYS

LL_RCC_PLL_GetMainSource

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_PLL_GetMainSource (void)</code>
Function description	Get the oscillator used as PLL clock source.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_PLLSOURCE_HSI - LL_RCC_PLLSOURCE_HSE
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CFGR PLLSRC LL_RCC_PLL_GetMainSource

LL_RCC_PLL_GetMultiplicator

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_PLL_GetMultiplicator (void)</code>
Function description	Get PLL multiplication Factor.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_PLL_MUL_3 - LL_RCC_PLL_MUL_4 - LL_RCC_PLL_MUL_6 - LL_RCC_PLL_MUL_8 - LL_RCC_PLL_MUL_12 - LL_RCC_PLL_MUL_16 - LL_RCC_PLL_MUL_24 - LL_RCC_PLL_MUL_32

- LL_RCC_PLL_MUL_48
 - CFGR PLLMUL LL_RCC_PLL_GetMultiplicator
- Reference Manual to
LL API cross
reference:

LL_RCC_PLL_GetDivider

- Function name **`__STATIC_INLINE uint32_t LL_RCC_PLL_GetDivider (void)`**
- Function description Get Division factor for the main PLL and other PLL.
- Return values
- **Returned:** value can be one of the following values:
 - LL_RCC_PLL_DIV_2
 - LL_RCC_PLL_DIV_3
 - LL_RCC_PLL_DIV_4
- Reference Manual to
LL API cross
reference:
- CFGR PLLDIV LL_RCC_PLL_GetDivider

LL_RCC_ClearFlag_LSIRDY

- Function name **`__STATIC_INLINE void LL_RCC_ClearFlag_LSIRDY (void)`**
- Function description Clear LSI ready interrupt flag.
- Return values
- **None:**
- Reference Manual to
LL API cross
reference:
- CIR LSIRDYC LL_RCC_ClearFlag_LSIRDY

LL_RCC_ClearFlag_LSERDY

- Function name **`__STATIC_INLINE void LL_RCC_ClearFlag_LSERDY (void)`**
- Function description Clear LSE ready interrupt flag.
- Return values
- **None:**
- Reference Manual to
LL API cross
reference:
- CIR LSERDYC LL_RCC_ClearFlag_LSERDY

LL_RCC_ClearFlag_MSIRDY

- Function name **`__STATIC_INLINE void LL_RCC_ClearFlag_MSIRDY (void)`**
- Function description Clear MSI ready interrupt flag.
- Return values
- **None:**
- Reference Manual to
LL API cross
reference:
- CIR MSIRDYC LL_RCC_ClearFlag_MSIRDY

LL_RCC_ClearFlag_HSIRDY

- Function name **`__STATIC_INLINE void LL_RCC_ClearFlag_HSIRDY (void)`**

Function description	Clear HSI ready interrupt flag.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR HSIRDYC LL_RCC_ClearFlag_HSIRDY

LL_RCC_ClearFlag_HSIRDY

Function name	<code>__STATIC_INLINE void LL_RCC_ClearFlag_HSIRDY (void)</code>
Function description	Clear HSE ready interrupt flag.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR HSERDYC LL_RCC_ClearFlag_HSERDY

LL_RCC_ClearFlag_PLLRDY

Function name	<code>__STATIC_INLINE void LL_RCC_ClearFlag_PLLRDY (void)</code>
Function description	Clear PLL ready interrupt flag.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR PLLRDYC LL_RCC_ClearFlag_PLLRDY

LL_RCC_ClearFlag_HSECSS

Function name	<code>__STATIC_INLINE void LL_RCC_ClearFlag_HSECSS (void)</code>
Function description	Clear Clock security system interrupt flag.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR CSSC LL_RCC_ClearFlag_HSECSS

LL_RCC_ClearFlag_LSECSS

Function name	<code>__STATIC_INLINE void LL_RCC_ClearFlag_LSECSS (void)</code>
Function description	Clear LSE Clock security system interrupt flag.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR LSECSSC LL_RCC_ClearFlag_LSECSS

LL_RCC_IsActiveFlag_LSIRDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LSIRDY (void)</code>
---------------	---

Function description	Check if LSI ready interrupt occurred or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR LSIRDYF LL_RCC_IsActiveFlag_LSIRDY

LL_RCC_IsActiveFlag_LSERDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LSERDY(void)</code>
Function description	Check if LSE ready interrupt occurred or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR LSERDYF LL_RCC_IsActiveFlag_LSERDY

LL_RCC_IsActiveFlag_MSIRDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_MSIRDY(void)</code>
Function description	Check if MSI ready interrupt occurred or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR MSIRDYF LL_RCC_IsActiveFlag_MSIRDY

LL_RCC_IsActiveFlag_HSIRDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSIRDY(void)</code>
Function description	Check if HSI ready interrupt occurred or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR HSIRDYF LL_RCC_IsActiveFlag_HSIRDY

LL_RCC_IsActiveFlag_HSERDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSERDY(void)</code>
Function description	Check if HSE ready interrupt occurred or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR HSERDYF LL_RCC_IsActiveFlag_HSERDY

LL_RCC_IsActiveFlag_PLLRDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PLLRDY(void)</code>
Function description	Check if PLL ready interrupt occurred or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR PLLRDYF LL_RCC_IsActiveFlag_PLLRDY

LL_RCC_IsActiveFlag_HSECSS

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSECSS(void)</code>
Function description	Check if Clock security system interrupt occurred or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR CSSF LL_RCC_IsActiveFlag_HSECSS

LL_RCC_IsActiveFlag_LSECSS

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LSECSS(void)</code>
Function description	Check if LSE Clock security system interrupt occurred or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR LSECSSF LL_RCC_IsActiveFlag_LSECSS

LL_RCC_IsActiveFlag_IWDGRST

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_IWDGRST(void)</code>
Function description	Check if RCC flag Independent Watchdog reset is set or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CSR IWDGRSTF LL_RCC_IsActiveFlag_IWDGRST

LL_RCC_IsActiveFlag_LPWRRST

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LPWRRST(void)</code>
Function description	Check if RCC flag Low Power reset is set or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

- Reference Manual to CSR LPWRRSTF LL_RCC_IsActiveFlag_LPWRRST
LL API cross reference:
- CSR LPWRRSTF LL_RCC_IsActiveFlag_LPWRRST

LL_RCC_IsActiveFlag_OBLRST

- Function name `__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_OBLRST(void)`
- Function description Check if RCC flag is set or not.
- Return values
 - State:** of bit (1 or 0).
- Reference Manual to CSR OBLRSTF LL_RCC_IsActiveFlag_OBLRST
LL API cross reference:

LL_RCC_IsActiveFlag_PINRST

- Function name `__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PINRST(void)`
- Function description Check if RCC flag Pin reset is set or not.
- Return values
 - State:** of bit (1 or 0).
- Reference Manual to CSR PINRSTF LL_RCC_IsActiveFlag_PINRST
LL API cross reference:

LL_RCC_IsActiveFlag_PORRST

- Function name `__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PORRST(void)`
- Function description Check if RCC flag POR/PDR reset is set or not.
- Return values
 - State:** of bit (1 or 0).
- Reference Manual to CSR PORRSTF LL_RCC_IsActiveFlag_PORRST
LL API cross reference:

LL_RCC_IsActiveFlag_SFTRST

- Function name `__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_SFTRST(void)`
- Function description Check if RCC flag Software reset is set or not.
- Return values
 - State:** of bit (1 or 0).
- Reference Manual to CSR SFTRSTF LL_RCC_IsActiveFlag_SFTRST
LL API cross reference:

LL_RCC_IsActiveFlag_WWDGRST

- Function name `__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_WWDGRST(void)`

Function description	Check if RCC flag Window Watchdog reset is set or not.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	CSR WWDGRSTF LL_RCC_IsActiveFlag_WWDGRST

LL_RCC_ClearResetFlags

Function name	<code>__STATIC_INLINE void LL_RCC_ClearResetFlags (void)</code>
Function description	Set RMVF bit to clear the reset flags.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	CSR RMVF LL_RCC_ClearResetFlags

LL_RCC_EnableIT_LSIRDY

Function name	<code>__STATIC_INLINE void LL_RCC_EnableIT_LSIRDY (void)</code>
Function description	Enable LSI ready interrupt.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	CIR LSIRDYIE LL_RCC_EnableIT_LSIRDY

LL_RCC_EnableIT_LSERDY

Function name	<code>__STATIC_INLINE void LL_RCC_EnableIT_LSERDY (void)</code>
Function description	Enable LSE ready interrupt.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	CIR LSERDYIE LL_RCC_EnableIT_LSERDY

LL_RCC_EnableIT_MSIRDY

Function name	<code>__STATIC_INLINE void LL_RCC_EnableIT_MSIRDY (void)</code>
Function description	Enable MSI ready interrupt.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	CIR MSIRDYIE LL_RCC_EnableIT_MSIRDY

LL_RCC_EnableIT_HSIRDY

Function name	<code>__STATIC_INLINE void LL_RCC_EnableIT_HSIRDY (void)</code>
Function description	Enable HSI ready interrupt.

Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR HSIRDYIE LL_RCC_EnableIT_HSIRDY

LL_RCC_EnableIT_HSIRDY

Function name	<u>__STATIC_INLINE void LL_RCC_EnableIT_HSIRDY (void)</u>
Function description	Enable HSE ready interrupt.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR HSERDYIE LL_RCC_EnableIT_HSIRDY

LL_RCC_EnableIT_PLLRDY

Function name	<u>__STATIC_INLINE void LL_RCC_EnableIT_PLLRDY (void)</u>
Function description	Enable PLL ready interrupt.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR PLLRDYIE LL_RCC_EnableIT_PLLRDY

LL_RCC_EnableIT_LSECSS

Function name	<u>__STATIC_INLINE void LL_RCC_EnableIT_LSECSS (void)</u>
Function description	Enable LSE clock security system interrupt.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR LSECSSIE LL_RCC_EnableIT_LSECSS

LL_RCC_DisableIT_LSIRDY

Function name	<u>__STATIC_INLINE void LL_RCC_DisableIT_LSIRDY (void)</u>
Function description	Disable LSI ready interrupt.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR LSIRDYIE LL_RCC_DisableIT_LSIRDY

LL_RCC_DisableIT_LSERDY

Function name	<u>__STATIC_INLINE void LL_RCC_DisableIT_LSERDY (void)</u>
Function description	Disable LSE ready interrupt.
Return values	<ul style="list-style-type: none"> • None:

- Reference Manual to • CIR LSERDYIE LL_RCC_DisableIT_LSERDY
LL API cross reference:

LL_RCC_DisableIT_MSIRDY

- Function name **_STATIC_INLINE void LL_RCC_DisableIT_MSIRDY (void)**
Function description Disable MSI ready interrupt.
Return values • **None:**
Reference Manual to • CIR MSIRDYIE LL_RCC_DisableIT_MSIRDY
LL API cross reference:

LL_RCC_DisableIT_HSIRDY

- Function name **_STATIC_INLINE void LL_RCC_DisableIT_HSIRDY (void)**
Function description Disable HSI ready interrupt.
Return values • **None:**
Reference Manual to • CIR HSIRDYIE LL_RCC_DisableIT_HSIRDY
LL API cross reference:

LL_RCC_DisableIT_HSERDY

- Function name **_STATIC_INLINE void LL_RCC_DisableIT_HSERDY (void)**
Function description Disable HSE ready interrupt.
Return values • **None:**
Reference Manual to • CIR HSERDYIE LL_RCC_DisableIT_HSERDY
LL API cross reference:

LL_RCC_DisableIT_PLLRDY

- Function name **_STATIC_INLINE void LL_RCC_DisableIT_PLLRDY (void)**
Function description Disable PLL ready interrupt.
Return values • **None:**
Reference Manual to • CIR PLLRDYIE LL_RCC_DisableIT_PLLRDY
LL API cross reference:

LL_RCC_DisableIT_LSECSS

- Function name **_STATIC_INLINE void LL_RCC_DisableIT_LSECSS (void)**
Function description Disable LSE clock security system interrupt.
Return values • **None:**
Reference Manual to • CIR LSECSSIE LL_RCC_DisableIT_LSECSS

LL API cross
reference:

LL_RCC_IsEnabledIT_LSIRDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_LSIRDY(void)</code>
Function description	Checks if LSI ready interrupt source is enabled or disabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR LSIRDYIE LL_RCC_IsEnabledIT_LSIRDY

LL_RCC_IsEnabledIT_LSERDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_LSERDY(void)</code>
Function description	Checks if LSE ready interrupt source is enabled or disabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR LSERDYIE LL_RCC_IsEnabledIT_LSERDY

LL_RCC_IsEnabledIT_MSIRDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_MSIRDY(void)</code>
Function description	Checks if MSI ready interrupt source is enabled or disabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR MSIRDYIE LL_RCC_IsEnabledIT_MSIRDY

LL_RCC_IsEnabledIT_HSIRDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_HSIRDY(void)</code>
Function description	Checks if HSI ready interrupt source is enabled or disabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CIR HSIRDYIE LL_RCC_IsEnabledIT_HSIRDY

LL_RCC_IsEnabledIT_HSERDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_HSERDY(void)</code>
---------------	--

Function description	Checks if HSE ready interrupt source is enabled or disabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR HSERDYIE LL_RCC_IsEnabledIT_HSERRDY

LL_RCC_IsEnabledIT_PLLRDY

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_PLLRDY((void)</code>
Function description	Checks if PLL ready interrupt source is enabled or disabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR PLLRDYIE LL_RCC_IsEnabledIT_PLLRDY

LL_RCC_IsEnabledIT_LSECSS

Function name	<code>__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_LSECSS((void)</code>
Function description	Checks if LSECSS interrupt source is enabled or disabled.
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	• CIR LSECSSIE LL_RCC_IsEnabledIT_LSECSS

LL_RCC_DeInit

Function name	ErrorStatus LL_RCC_DeInit (void)
Function description	Reset the RCC clock configuration to the default reset state.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: RCC registers are de-initialized – ERROR: not applicable
Notes	<ul style="list-style-type: none"> • The default reset state of the clock configuration is given below: MSI ON and used as system clock sourceHSE, HSI and PLL OFFAHB, APB1 and APB2 prescaler set to 1.CSS, MCO OFFAll interrupts disabled • This function doesn't modify the configuration of the Peripheral clocksLSI, LSE and RTC clocks

LL_RCC_GetSystemClocksFreq

Function name	<code>void LL_RCC_GetSystemClocksFreq((LL_RCC_ClocksTypeDef * RCC_Clocks)</code>
Function description	Return the frequencies of different on chip clocks; System, AHB, APB1 and APB2 buses clocks.
Parameters	<ul style="list-style-type: none"> • RCC_Clocks: pointer to a LL_RCC_ClocksTypeDef

structure which will hold the clocks frequencies

- | | |
|---------------|---|
| Return values | <ul style="list-style-type: none"> • None: |
| Notes | <ul style="list-style-type: none"> • Each time SYSCLK, HCLK, PCLK1 and/or PCLK2 clock changes, this function must be called to update structure fields. Otherwise, any configuration based on this function will be incorrect. |

62.3 RCC Firmware driver defines

62.3.1 RCC

APB low-speed prescaler (APB1)

LL_RCC_APB1_DIV_1	HCLK not divided
LL_RCC_APB1_DIV_2	HCLK divided by 2
LL_RCC_APB1_DIV_4	HCLK divided by 4
LL_RCC_APB1_DIV_8	HCLK divided by 8
LL_RCC_APB1_DIV_16	HCLK divided by 16

APB high-speed prescaler (APB2)

LL_RCC_APB2_DIV_1	HCLK not divided
LL_RCC_APB2_DIV_2	HCLK divided by 2
LL_RCC_APB2_DIV_4	HCLK divided by 4
LL_RCC_APB2_DIV_8	HCLK divided by 8
LL_RCC_APB2_DIV_16	HCLK divided by 16

Clear Flags Defines

LL_RCC_CIR_LSIRDYC	LSI Ready Interrupt Clear
LL_RCC_CIR_LSERDYC	LSE Ready Interrupt Clear
LL_RCC_CIR_HSIRDYC	HSI Ready Interrupt Clear
LL_RCC_CIR_HSERDYC	HSE Ready Interrupt Clear
LL_RCC_CIR_PLLRDYC	PLL Ready Interrupt Clear
LL_RCC_CIR_MSIRDYC	MSI Ready Interrupt Clear
LL_RCC_CIR_LSECSSC	LSE Clock Security System Interrupt Clear
LL_RCC_CIR_CSSC	Clock Security System Interrupt Clear

Get Flags Defines

LL_RCC_CIR_LSIRDYF	LSI Ready Interrupt flag
LL_RCC_CIR_LSERDYF	LSE Ready Interrupt flag
LL_RCC_CIR_HSIRDYF	HSI Ready Interrupt flag
LL_RCC_CIR_HSERDYF	HSE Ready Interrupt flag
LL_RCC_CIR_PLLRDYF	PLL Ready Interrupt flag
LL_RCC_CIR_MSIRDYF	MSI Ready Interrupt flag

LL_RCC_CIR_LSECSSF	LSE Clock Security System Interrupt flag
LL_RCC_CIR_CSSF	Clock Security System Interrupt flag
LL_RCC_CSR_OBLRSTF	OBL reset flag
LL_RCC_CSR_PINRSTF	PIN reset flag
LL_RCC_CSR_PORRSTF	POR/PDR reset flag
LL_RCC_CSR_SFTRSTF	Software Reset flag
LL_RCC_CSR_IWDGRSTF	Independent Watchdog reset flag
LL_RCC_CSR_WWDGRSTF	Window watchdog reset flag
LL_RCC_CSR_LPWRRSTF	Low-Power reset flag

IT Defines

LL_RCC_CIR_LSIRDYIE	LSI Ready Interrupt Enable
LL_RCC_CIR_LSERDYIE	LSE Ready Interrupt Enable
LL_RCC_CIR_HSIRDYIE	HSI Ready Interrupt Enable
LL_RCC_CIR_HSERDYIE	HSE Ready Interrupt Enable
LL_RCC_CIR_PLLRDYIE	PLL Ready Interrupt Enable
LL_RCC_CIR_MSIRDYIE	MSI Ready Interrupt Enable
LL_RCC_CIR_LSECSSIE	LSE CSS Interrupt Enable

MCO1 SOURCE selection

LL_RCC_MCO1SOURCE_NOCLOCK	MCO output disabled, no clock on MCO
LL_RCC_MCO1SOURCE_SYSCLK	SYSCLK selection as MCO source
LL_RCC_MCO1SOURCE_HSI	HSI selection as MCO source
LL_RCC_MCO1SOURCE_MSI	MSI selection as MCO source
LL_RCC_MCO1SOURCE_HSE	HSE selection as MCO source
LL_RCC_MCO1SOURCE_LSI	LSI selection as MCO source
LL_RCC_MCO1SOURCE_LSE	LSE selection as MCO source
LL_RCC_MCO1SOURCE_PLLCLK	PLLCLK selection as MCO source

MCO1 prescaler

LL_RCC_MCO1_DIV_1	MCO Clock divided by 1
LL_RCC_MCO1_DIV_2	MCO Clock divided by 2
LL_RCC_MCO1_DIV_4	MCO Clock divided by 4
LL_RCC_MCO1_DIV_8	MCO Clock divided by 8
LL_RCC_MCO1_DIV_16	MCO Clock divided by 16

MSI clock ranges

LL_RCC_MSIRANGE_0	MSI = 65.536 KHz
LL_RCC_MSIRANGE_1	MSI = 131.072 KHz
LL_RCC_MSIRANGE_2	MSI = 262.144 KHz

<code>LL_RCC_MSIRANGE_3</code>	MSI = 524.288 KHz
<code>LL_RCC_MSIRANGE_4</code>	MSI = 1.048 MHz
<code>LL_RCC_MSIRANGE_5</code>	MSI = 2.097 MHz
<code>LL_RCC_MSIRANGE_6</code>	MSI = 4.194 MHz

Oscillator Values adaptation

<code>HSE_VALUE</code>	Value of the HSE oscillator in Hz
<code>HSI_VALUE</code>	Value of the HSI oscillator in Hz
<code>LSE_VALUE</code>	Value of the LSE oscillator in Hz
<code>LSI_VALUE</code>	Value of the LSI oscillator in Hz

Peripheral clock frequency

<code>LL_RCC_PERIPH_FREQUENCY_NO</code>	No clock enabled for the peripheral
<code>LL_RCC_PERIPH_FREQUENCY_NA</code>	Frequency cannot be provided as external clock

PLL SOURCE

<code>LL_RCC_PLLSOURCE_HSI</code>	HSI clock selected as PLL entry clock source
<code>LL_RCC_PLLSOURCE_HSE</code>	HSE clock selected as PLL entry clock source

PLL division factor

<code>LL_RCC_PLL_DIV_2</code>	PLL clock output = PLLVCO / 2
<code>LL_RCC_PLL_DIV_3</code>	PLL clock output = PLLVCO / 3
<code>LL_RCC_PLL_DIV_4</code>	PLL clock output = PLLVCO / 4

PLL Multiplicator factor

<code>LL_RCC_PLL_MUL_3</code>	PLL input clock * 3
<code>LL_RCC_PLL_MUL_4</code>	PLL input clock * 4
<code>LL_RCC_PLL_MUL_6</code>	PLL input clock * 6
<code>LL_RCC_PLL_MUL_8</code>	PLL input clock * 8
<code>LL_RCC_PLL_MUL_12</code>	PLL input clock * 12
<code>LL_RCC_PLL_MUL_16</code>	PLL input clock * 16
<code>LL_RCC_PLL_MUL_24</code>	PLL input clock * 24
<code>LL_RCC_PLL_MUL_32</code>	PLL input clock * 32
<code>LL_RCC_PLL_MUL_48</code>	PLL input clock * 48

RTC clock source selection

<code>LL_RCC_RTC_CLKSOURCE_NONE</code>	No clock used as RTC clock
<code>LL_RCC_RTC_CLKSOURCE_LSE</code>	LSE oscillator clock used as RTC clock
<code>LL_RCC_RTC_CLKSOURCE_LSI</code>	LSI oscillator clock used as RTC clock
<code>LL_RCC_RTC_CLKSOURCE_HSE</code>	HSE oscillator clock divided by a programmable prescaler (selection through

RTC HSE Prescaler

<code>LL_RCC_RTC_HSE_DIV_2</code>	HSE is divided by 2 for RTC clock
-----------------------------------	-----------------------------------

LL_RCC_RTC_HSE_DIV_4	HSE is divided by 4 for RTC clock
LL_RCC_RTC_HSE_DIV_8	HSE is divided by 8 for RTC clock
LL_RCC_RTC_HSE_DIV_16	HSE is divided by 16 for RTC clock

AHB prescaler

LL_RCC_SYSCLK_DIV_1	SYSCLK not divided
LL_RCC_SYSCLK_DIV_2	SYSCLK divided by 2
LL_RCC_SYSCLK_DIV_4	SYSCLK divided by 4
LL_RCC_SYSCLK_DIV_8	SYSCLK divided by 8
LL_RCC_SYSCLK_DIV_16	SYSCLK divided by 16
LL_RCC_SYSCLK_DIV_64	SYSCLK divided by 64
LL_RCC_SYSCLK_DIV_128	SYSCLK divided by 128
LL_RCC_SYSCLK_DIV_256	SYSCLK divided by 256
LL_RCC_SYSCLK_DIV_512	SYSCLK divided by 512

System clock switch

LL_RCC_SYS_CLKSOURCE_MSI	MSI selection as system clock
LL_RCC_SYS_CLKSOURCE_HSI	HSI selection as system clock
LL_RCC_SYS_CLKSOURCE_HSE	HSE selection as system clock
LL_RCC_SYS_CLKSOURCE_PLL	PLL selection as system clock

System clock switch status

LL_RCC_SYS_CLKSOURCE_STATUS_MSI	MSI used as system clock
LL_RCC_SYS_CLKSOURCE_STATUS_HSI	HSI used as system clock
LL_RCC_SYS_CLKSOURCE_STATUS_HSE	HSE used as system clock
LL_RCC_SYS_CLKSOURCE_STATUS_PLL	PLL used as system clock

Calculate frequencies

<u>LL_RCC_CALC_PLLCLK_F</u> REQ	Description:
	<ul style="list-style-type: none"> Helper macro to calculate the PLLCLK frequency.
	Parameters:
	<ul style="list-style-type: none"> <u>INPUTFREQ</u>: PLL Input frequency (based on MSI/HSE/HSI) <u>PLLMUL</u>: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RCC_PLL_MUL_3 - LL_RCC_PLL_MUL_4 - LL_RCC_PLL_MUL_6 - LL_RCC_PLL_MUL_8 - LL_RCC_PLL_MUL_12 - LL_RCC_PLL_MUL_16 - LL_RCC_PLL_MUL_24 - LL_RCC_PLL_MUL_32 - LL_RCC_PLL_MUL_48 <u>PLLDIV</u>: This parameter can be one of the

following values:

- LL_RCC_PLL_DIV_2
- LL_RCC_PLL_DIV_3
- LL_RCC_PLL_DIV_4

Return value:

- PLL: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLCLK_FREQ
(HSE_VALUE, LL_RCC_PLL_GetMultiplicator (),
LL_RCC_PLL_GetDivider ());

__LL_RCC_CALC_HCLK_FR
EQ

Description:

- Helper macro to calculate the HCLK frequency.

Parameters:

- __SYSCLKFREQ__: SYSCLK frequency (based on MSI/HSE/HSI/PLLCLK)
- __AHBPRESCALER__: This parameter can be one of the following values:
 - LL_RCC_SYSCLK_DIV_1
 - LL_RCC_SYSCLK_DIV_2
 - LL_RCC_SYSCLK_DIV_4
 - LL_RCC_SYSCLK_DIV_8
 - LL_RCC_SYSCLK_DIV_16
 - LL_RCC_SYSCLK_DIV_64
 - LL_RCC_SYSCLK_DIV_128
 - LL_RCC_SYSCLK_DIV_256
 - LL_RCC_SYSCLK_DIV_512

Return value:

- HCLK: clock frequency (in Hz)

Notes:

- : __AHBPRESCALER__ be retrieved by
LL_RCC_GetAHBPrescaler ex:
__LL_RCC_CALC_HCLK_FREQ(LL_RCC_GetAHB
Prescaler())

__LL_RCC_CALC_PCLK1_F
REQ

Description:

- Helper macro to calculate the PCLK1 frequency
(APB1)

Parameters:

- __HCLKFREQ__: HCLK frequency
- __APB1PRESCALER__: This parameter can be one of the following values:
 - LL_RCC_APB1_DIV_1
 - LL_RCC_APB1_DIV_2
 - LL_RCC_APB1_DIV_4
 - LL_RCC_APB1_DIV_8
 - LL_RCC_APB1_DIV_16

Return value:

- PCLK1: clock frequency (in Hz)

Notes:

- : __APB1PRESCALER__ be retrieved by
LL_RCC_GetAPB1Prescaler ex:
__LL_RCC_CALC_PCLK1_FREQ(LL_RCC_GetAPB1Prescaler())

`__LL_RCC_CALC_PCLK2_FREQ`

Description:

- Helper macro to calculate the PCLK2 frequency (ABP2)

Parameters:

- __HCLKFREQ__: HCLK frequency
- __APB2PRESCALER__: This parameter can be one of the following values:
 - LL_RCC_APB2_DIV_1
 - LL_RCC_APB2_DIV_2
 - LL_RCC_APB2_DIV_4
 - LL_RCC_APB2_DIV_8
 - LL_RCC_APB2_DIV_16

Return value:

- PCLK2: clock frequency (in Hz)

Notes:

- : __APB2PRESCALER__ be retrieved by
LL_RCC_GetAPB2Prescaler ex:
__LL_RCC_CALC_PCLK2_FREQ(LL_RCC_GetAPB2Prescaler())

`__LL_RCC_CALC_MSI_FRE_Q`

Description:

- Helper macro to calculate the MSI frequency (in Hz)

Parameters:

- __MSIRANGE__: This parameter can be one of the following values:
 - LL_RCC_MSIRANGE_0
 - LL_RCC_MSIRANGE_1
 - LL_RCC_MSIRANGE_2
 - LL_RCC_MSIRANGE_3
 - LL_RCC_MSIRANGE_4
 - LL_RCC_MSIRANGE_5
 - LL_RCC_MSIRANGE_6

Return value:

- MSI: clock frequency (in Hz)

Notes:

- : __MSIRANGE__ can be retrieved by
LL_RCC_MSI_GetRange ex:
__LL_RCC_CALC_MSI_FREQ(LL_RCC_MSI_GetR

ange())

Common Write and read registers Macros

LL_RCC_WriteReg Description:

- Write a value in RCC register.

Parameters:

- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_RCC_ReadReg Description:

- Read a value in RCC register.

Parameters:

- __REG__: Register to be read

Return value:

- Register: value

63 LL RTC Generic Driver

63.1 RTC Firmware driver registers structures

63.1.1 LL_RTC_InitTypeDef

Data Fields

- *uint32_t HourFormat*
- *uint32_t AsynchPrescaler*
- *uint32_t SynchPrescaler*

Field Documentation

- ***uint32_t LL_RTC_InitTypeDef::HourFormat***
Specifies the RTC Hours Format. This parameter can be a value of **RTC_LL_EC_HOURFORMAT**This feature can be modified afterwards using unitary function **LL_RTC_SetHourFormat()**.
- ***uint32_t LL_RTC_InitTypeDef::AsynchPrescaler***
Specifies the RTC Asynchronous Predivider value. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x7FThis feature can be modified afterwards using unitary function **LL_RTC_SetAsynchPrescaler()**.
- ***uint32_t LL_RTC_InitTypeDef::SynchPrescaler***
Specifies the RTC Synchronous Predivider value. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x7FFFThis feature can be modified afterwards using unitary function **LL_RTC_SetSynchPrescaler()**.

63.1.2 LL_RTC_TimeTypeDef

Data Fields

- *uint32_t TimeFormat*
- *uint8_t Hours*
- *uint8_t Minutes*
- *uint8_t Seconds*

Field Documentation

- ***uint32_t LL_RTC_TimeTypeDef::TimeFormat***
Specifies the RTC AM/PM Time. This parameter can be a value of **RTC_LL_EC_TIME_FORMAT**This feature can be modified afterwards using unitary function **LL_RTC_TIME_SetFormat()**.
- ***uint8_t LL_RTC_TimeTypeDef::Hours***
Specifies the RTC Time Hours. This parameter must be a number between Min_Data = 0 and Max_Data = 12 if the **LL_RTC_TIME_FORMAT_PM** is selected. This parameter must be a number between Min_Data = 0 and Max_Data = 23 if the **LL_RTC_TIME_FORMAT_AM_OR_24** is selected.This feature can be modified afterwards using unitary function **LL_RTC_TIME_SetHour()**.
- ***uint8_t LL_RTC_TimeTypeDef::Minutes***
Specifies the RTC Time Minutes. This parameter must be a number between Min_Data = 0 and Max_Data = 59This feature can be modified afterwards using unitary function **LL_RTC_TIME_SetMinute()**.
- ***uint8_t LL_RTC_TimeTypeDef::Seconds***
Specifies the RTC Time Seconds. This parameter must be a number between

Min_Data = 0 and Max_Data = 59This feature can be modified afterwards using unitary function **LL_RTC_TIME_SetSecond()**.

63.1.3 LL_RTC_DateTypeDef

Data Fields

- *uint8_t WeekDay*
- *uint8_t Month*
- *uint8_t Day*
- *uint8_t Year*

Field Documentation

- ***uint8_t LL_RTC_DateTypeDef::WeekDay***
Specifies the RTC Date WeekDay. This parameter can be a value of **RTC_LL_EC_WEEKDAY**This feature can be modified afterwards using unitary function **LL_RTC_DATE_SetWeekDay()**.
- ***uint8_t LL_RTC_DateTypeDef::Month***
Specifies the RTC Date Month. This parameter can be a value of **RTC_LL_EC_MONTH**This feature can be modified afterwards using unitary function **LL_RTC_DATE_SetMonth()**.
- ***uint8_t LL_RTC_DateTypeDef::Day***
Specifies the RTC Date Day. This parameter must be a number between Min_Data = 1 and Max_Data = 31This feature can be modified afterwards using unitary function **LL_RTC_DATE_SetDay()**.
- ***uint8_t LL_RTC_DateTypeDef::Year***
Specifies the RTC Date Year. This parameter must be a number between Min_Data = 0 and Max_Data = 99This feature can be modified afterwards using unitary function **LL_RTC_DATE_SetYear()**.

63.1.4 LL_RTC_AlarmTypeDef

Data Fields

- ***LL_RTC_TimeTypeDef AlarmTime***
- ***uint32_t AlarmMask***
- ***uint32_t AlarmDateWeekDaySel***
- ***uint8_t AlarmDateWeekDay***

Field Documentation

- ***LL_RTC_TimeTypeDef LL_RTC_AlarmTypeDef::AlarmTime***
Specifies the RTC Alarm Time members.
- ***uint32_t LL_RTC_AlarmTypeDef::AlarmMask***
Specifies the RTC Alarm Masks. This parameter can be a value of **RTC_LL_EC_ALMA_MASK** for ALARM A or **RTC_LL_EC_ALMB_MASK** for ALARM B.This feature can be modified afterwards using unitary function **LL_RTC_ALMA_SetMask()** for ALARM A or **LL_RTC_ALMB_SetMask()** for ALARM B
- ***uint32_t LL_RTC_AlarmTypeDef::AlarmDateWeekDaySel***
Specifies the RTC Alarm is on day or WeekDay. This parameter can be a value of **RTC_LL_EC_ALMA_WEEKDAY_SELECTION** for ALARM A or **RTC_LL_EC_ALMB_WEEKDAY_SELECTION** for ALARM BThis feature can be modified afterwards using unitary function **LL_RTC_ALMA_EnableWeekday()** or **LL_RTC_ALMA_DisableWeekday()** for ALARM A or **LL_RTC_ALMB_EnableWeekday()** or **LL_RTC_ALMB_DisableWeekday()** for ALARM B

- ***uint8_t LL_RTC_AlarmTypeDef::AlarmDateWeekDay***

Specifies the RTC Alarm Day/WeekDay. If AlarmDateWeekDaySel set to day, this parameter must be a number between Min_Data = 1 and Max_Data = 31. This feature can be modified afterwards using unitary function **LL_RTC_ALMA_SetDay()** for ALARM A or **LL_RTC_ALMB_SetDay()** for ALARM B. If AlarmDateWeekDaySel set to Weekday, this parameter can be a value of **RTC_LL_EC_WEEKDAY**. This feature can be modified afterwards using unitary function **LL_RTC_ALMA_SetWeekDay()** for ALARM A or **LL_RTC_ALMB_SetWeekDay()** for ALARM B.

63.2 RTC Firmware driver API description

63.2.1 Detailed description of functions

LL_RTC_SetHourFormat

Function name	<code>_STATIC_INLINE void LL_RTC_SetHourFormat (RTC_TypeDef * RTCx, uint32_t HourFormat)</code>
Function description	Set Hours format (24 hour/day or AM/PM hour format)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • HourFormat: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_HOURFORMAT_24HOUR – LL_RTC_HOURFORMAT_AMPM
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. <code>LL_RTC_DisableWriteProtection</code> function should be called before. • It can be written in initialization mode only (<code>LL_RTC_EnableInitMode</code> function)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR FMT <code>LL_RTC_SetHourFormat</code>

LL_RTC_GetHourFormat

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_GetHourFormat (RTC_TypeDef * RTCx)</code>
Function description	Get Hours format (24 hour/day or AM/PM hour format)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_HOURFORMAT_24HOUR – LL_RTC_HOURFORMAT_AMPM
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR FMT <code>LL_RTC_GetHourFormat</code>

LL_RTC_SetAlarmOutEvent

Function name	<code>_STATIC_INLINE void LL_RTC_SetAlarmOutEvent (RTC_TypeDef * RTCx, uint32_t AlarmOutput)</code>
---------------	--

Function description	Select the flag to be routed to RTC_ALARM output.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance AlarmOutput: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_ALARMOUT_DISABLE – LL_RTC_ALARMOUT_ALMA – LL_RTC_ALARMOUT_ALMB – LL_RTC_ALARMOUT_WAKEUP
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR OSEL LL_RTC_SetAlarmOutEvent

LL_RTC_GetAlarmOutEvent

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_GetAlarmOutEvent((RTC_TypeDef * RTCx)</code>
Function description	Get the flag to be routed to RTC_ALARM output.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_ALARMOUT_DISABLE – LL_RTC_ALARMOUT_ALMA – LL_RTC_ALARMOUT_ALMB – LL_RTC_ALARMOUT_WAKEUP
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR OSEL LL_RTC_GetAlarmOutEvent

LL_RTC_SetAlarmOutputType

Function name	<code>__STATIC_INLINE void LL_RTC_SetAlarmOutputType((RTC_TypeDef * RTCx, uint32_t Output)</code>
Function description	Set RTC_ALARM output type (ALARM in push-pull or open-drain output)
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Output: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_ALARM_OUTPUTTYPE_OPENDRAIN – LL_RTC_ALARM_OUTPUTTYPE_PUSH_PULL
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Used only when RTC_ALARM is mapped on PC13
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> TAFCR ALARMOUTTYPE LL_RTC_SetAlarmOutputType

LL_RTC_GetAlarmOutputType

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_GetAlarmOutputType (RTC_TypeDef * RTCx)</code>
Function description	Get RTC_ALARM output type (ALARM in push-pull or open-drain output)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_ALARM_OUTPUTTYPE_OPENDRAIN – LL_RTC_ALARM_OUTPUTTYPE_PUSH_PULL
Notes	<ul style="list-style-type: none"> • used only when RTC_ALARM is mapped on PC13
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR ALARMOUTTYPE LL_RTC_GetAlarmOutputType

LL_RTC_EnableInitMode

Function name	<code>_STATIC_INLINE void LL_RTC_EnableInitMode (RTC_TypeDef * RTCx)</code>
Function description	Enable initialization mode.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Initialization mode is used to program time and date register (RTC_TR and RTC_DR) and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR INIT LL_RTC_EnableInitMode

LL_RTC_DisableInitMode

Function name	<code>_STATIC_INLINE void LL_RTC_DisableInitMode (RTC_TypeDef * RTCx)</code>
Function description	Disable initialization mode (Free running mode)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:

Reference Manual to
LL API cross
reference:

- ISR INIT LL_RTC_DisableInitMode

LL_RTC_SetOutputPolarity

Function name	<code>_STATIC_INLINE void LL_RTC_SetOutputPolarity (RTC_TypeDef * RTCx, uint32_t Polarity)</code>
Function description	Set Output polarity (pin is low when ALRAF/ALRBF/WUTF is

	asserted)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Polarity: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_OUTPUTPOLARITY_PIN_HIGH – LL_RTC_OUTPUTPOLARITY_PIN_LOW
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR POL LL_RTC_SetOutputPolarity

LL_RTC_GetOutputPolarity

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_GetOutputPolarity((RTC_TypeDef * RTCx))</code>
Function description	Get Output polarity.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_OUTPUTPOLARITY_PIN_HIGH – LL_RTC_OUTPUTPOLARITY_PIN_LOW
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR POL LL_RTC_GetOutputPolarity

LL_RTC_EnableShadowRegBypass

Function name	<code>__STATIC_INLINE void LL_RTC_EnableShadowRegBypass((RTC_TypeDef * RTCx))</code>
Function description	Enable Bypass the shadow registers.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR BYPSHAD LL_RTC_EnableShadowRegBypass

LL_RTC_DisableShadowRegBypass

Function name	<code>__STATIC_INLINE void LL_RTC_DisableShadowRegBypass((RTC_TypeDef * RTCx))</code>
Function description	Disable Bypass the shadow registers.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:

- Reference Manual to LL API cross reference:
- CR BYPSHAD LL_RTC_DisableShadowRegBypass

LL_RTC_IsShadowRegBypassEnabled

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsShadowRegBypassEnabled (RTC_TypeDef * RTCx)</code>
Function description	Check if Shadow registers bypass is enabled or not.
Parameters	<ul style="list-style-type: none">• RTCx: RTC Instance
Return values	<ul style="list-style-type: none">• State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR BYPSHAD LL_RTC_IsShadowRegBypassEnabled

LL_RTC_EnableRefClock

Function name	<code>__STATIC_INLINE void LL_RTC_EnableRefClock (RTC_TypeDef * RTCx)</code>
Function description	Enable RTC_REFIN reference clock detection (50 or 60 Hz)
Parameters	<ul style="list-style-type: none">• RTCx: RTC Instance
Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.• It can be written in initialization mode only (LL_RTC_EnableInitMode function)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR REFCKON LL_RTC_EnableRefClock

LL_RTC_DisableRefClock

Function name	<code>__STATIC_INLINE void LL_RTC_DisableRefClock (RTC_TypeDef * RTCx)</code>
Function description	Disable RTC_REFIN reference clock detection (50 or 60 Hz)
Parameters	<ul style="list-style-type: none">• RTCx: RTC Instance
Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.• It can be written in initialization mode only (LL_RTC_EnableInitMode function)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR REFCKON LL_RTC_DisableRefClock

LL_RTC_SetAsynchPrescaler

Function name **STATIC_INLINE void LL_RTC_SetAsynchPrescaler(RTC_TypeDef * RTCx, uint32_t AsynchPrescaler)**

Function description Set Asynchronous prescaler factor.

Parameters

- **RTCx:** RTC Instance
- **AsynchPrescaler:** Value between Min_Data = 0 and Max_Data = 0x7F

Return values

- **None:**

Reference Manual to
LL API cross
reference:
[PRER PREDIV_A LL_RTC_SetAsynchPrescaler](#)

LL_RTC_SetSynchPrescaler

Function name **STATIC_INLINE void LL_RTC_SetSynchPrescaler(RTC_TypeDef * RTCx, uint32_t SynchPrescaler)**

Function description Set Synchronous prescaler factor.

Parameters

- **RTCx:** RTC Instance
- **SynchPrescaler:** Value between Min_Data = 0 and Max_Data = 0x7FFF

Return values

- **None:**

Reference Manual to
LL API cross
reference:
[PRER PREDIV_S LL_RTC_SetSynchPrescaler](#)

LL_RTC_GetAsynchPrescaler

Function name **STATIC_INLINE uint32_t LL_RTC_GetAsynchPrescaler(RTC_TypeDef * RTCx)**

Function description Get Asynchronous prescaler factor.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data = 0 and Max_Data = 0x7F

Reference Manual to
LL API cross
reference:
[PRER PREDIV_A LL_RTC_GetAsynchPrescaler](#)

LL_RTC_GetSynchPrescaler

Function name **STATIC_INLINE uint32_t LL_RTC_GetSynchPrescaler(RTC_TypeDef * RTCx)**

Function description Get Synchronous prescaler factor.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data = 0 and Max_Data = 0x7FFF

Reference Manual to
LL API cross
reference:
[PRER PREDIV_S LL_RTC_GetSynchPrescaler](#)

reference:

LL_RTC_EnableWriteProtection

Function name	<code>__STATIC_INLINE void LL_RTC_EnableWriteProtection(RTC_TypeDef * RTCx)</code>
Function description	Enable the write protection for RTC registers.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • WPR KEY LL_RTC_EnableWriteProtection

LL_RTC_DisableWriteProtection

Function name	<code>__STATIC_INLINE void LL_RTC_DisableWriteProtection(RTC_TypeDef * RTCx)</code>
Function description	Disable the write protection for RTC registers.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • WPR KEY LL_RTC_DisableWriteProtection

LL_RTC_TIME_SetFormat

Function name	<code>__STATIC_INLINE void LL_RTC_TIME_SetFormat(RTC_TypeDef * RTCx, uint32_t TimeFormat)</code>
Function description	Set time format (AM/24-hour or PM notation)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • TimeFormat: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TIME_FORMAT_AM_OR_24 - LL_RTC_TIME_FORMAT_PM
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. • It can be written in initialization mode only (LL_RTC_EnableInitMode function)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TR PM LL_RTC_TIME_SetFormat

LL_RTC_TIME_GetFormat

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TIME_GetFormat(RTC_TypeDef * RTCx)</code>
---------------	--

Function description	Get time format (AM or PM notation)
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_TIME_FORMAT_AM_OR_24 – LL_RTC_TIME_FORMAT_PM
Notes	<ul style="list-style-type: none"> if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> TR PM LL_RTC_TIME_GetFormat

LL_RTC_TIME_SetHour

Function name	<code>__STATIC_INLINE void LL_RTC_TIME_SetHour (RTC_TypeDef * RTCx, uint32_t Hours)</code>
Function description	Set Hours in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Hours: Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. It can be written in initialization mode only (LL_RTC_EnableInitMode function) helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert hour from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> TR HT LL_RTC_TIME_SetHour TR HU LL_RTC_TIME_SetHour

LL_RTC_TIME_GetHour

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TIME_GetHour (RTC_TypeDef * RTCx)</code>
Function description	Get Hours in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23
Notes	<ul style="list-style-type: none"> if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)). helper macro __LL_RTC_CONVERT_BCD2BIN is available

to convert hour from BCD to Binary format

Reference Manual to
LL API cross
reference:

- TR HT LL_RTC_TIME_GetHour
- TR HU LL_RTC_TIME_GetHour

LL_RTC_TIME_SetMinute

Function name

**`_STATIC_INLINE void LL_RTC_TIME_SetMinute
(RTC_TypeDef * RTCx, uint32_t Minutes)`**

Function description

Set Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Minutes:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)
- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Minutes from binary to BCD format

Reference Manual to
LL API cross
reference:

- TR MNT LL_RTC_TIME_SetMinute
- TR MNU LL_RTC_TIME_SetMinute

LL_RTC_TIME_GetMinute

Function name

**`_STATIC_INLINE uint32_t LL_RTC_TIME_GetMinute
(RTC_TypeDef * RTCx)`**

Function description

Get Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)).
- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert minute from BCD to Binary format

Reference Manual to
LL API cross
reference:

- TR MNT LL_RTC_TIME_GetMinute
- TR MNU LL_RTC_TIME_GetMinute

LL_RTC_TIME_SetSecond

Function name

**`_STATIC_INLINE void LL_RTC_TIME_SetSecond
(RTC_TypeDef * RTCx, uint32_t Seconds)`**

Function description

Set Seconds in BCD format.

Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Seconds: Value between Min_Data=0x00 and Max_Data=0x59
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. It can be written in initialization mode only (LL_RTC_EnableInitMode function) helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Seconds from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> TR ST LL_RTC_TIME_SetSecond TR SU LL_RTC_TIME_SetSecond

LL_RTC_TIME_GetSecond

Function name	<code>STATIC_INLINE uint32_t LL_RTC_TIME_GetSecond (RTC_TypeDef * RTCx)</code>
Function description	Get Seconds in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x00 and Max_Data=0x59
Notes	<ul style="list-style-type: none"> if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)). helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Seconds from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> TR ST LL_RTC_TIME_GetSecond TR SU LL_RTC_TIME_GetSecond

LL_RTC_TIME_Config

Function name	<code>STATIC_INLINE void LL_RTC_TIME_Config (RTC_TypeDef * RTCx, uint32_t Format12_24, uint32_t Hours, uint32_t Minutes, uint32_t Seconds)</code>
Function description	Set time (hour, minute and second) in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Format12_24: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_TIME_FORMAT_AM_OR_24 – LL_RTC_TIME_FORMAT_PM Hours: Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23 Minutes: Value between Min_Data=0x00 and Max_Data=0x59 Seconds: Value between Min_Data=0x00 and

	Max_Data=0x59
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. • It can be written in initialization mode only (LL_RTC_EnableInitMode function) • TimeFormat and Hours should follow the same format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TR PM LL_RTC_TIME_Config • TR HT LL_RTC_TIME_Config • TR HU LL_RTC_TIME_Config • TR MNT LL_RTC_TIME_Config • TR MNU LL_RTC_TIME_Config • TR ST LL_RTC_TIME_Config • TR SU LL_RTC_TIME_Config

LL_RTC_TIME_Get

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TIME_Get (RTC_TypeDef * RTCx)</code>
Function description	Get time (hour, minute and second) in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Combination: of hours, minutes and seconds (Format: 0x00HHMMSS).
Notes	<ul style="list-style-type: none"> • if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit • Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)). • helper macros <code>_LL_RTC_GET_HOUR</code>, <code>_LL_RTC_GET_MINUTE</code> and <code>_LL_RTC_GET_SECOND</code> are available to get independently each parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TR HT LL_RTC_TIME_Get • TR HU LL_RTC_TIME_Get • TR MNT LL_RTC_TIME_Get • TR MNU LL_RTC_TIME_Get • TR ST LL_RTC_TIME_Get • TR SU LL_RTC_TIME_Get

LL_RTC_TIME_EnableDayLightStore

Function name	<code>__STATIC_INLINE void LL_RTC_TIME_EnableDayLightStore (RTC_TypeDef * RTCx)</code>
Function description	Memorize whether the daylight saving time change has been performed.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection

function should be called before.

Reference Manual to
LL API cross
reference:

- CR BKP LL_RTC_TIME_EnableDayLightStore

LL_RTC_TIME_DisableDayLightStore

Function name	<code>__STATIC_INLINE void LL_RTC_TIME_DisableDayLightStore (RTC_TypeDef * RTCx)</code>
Function description	Disable memorization whether the daylight saving time change has been performed.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR BKP LL_RTC_TIME_DisableDayLightStore

LL_RTC_TIME_IsDayLightStoreEnabled

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TIME_IsDayLightStoreEnabled (RTC_TypeDef * RTCx)</code>
Function description	Check if RTC Day Light Saving stored operation has been enabled or not.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR BKP LL_RTC_TIME_IsDayLightStoreEnabled

LL_RTC_TIME_DecHour

Function name	<code>__STATIC_INLINE void LL_RTC_TIME_DecHour (RTC_TypeDef * RTCx)</code>
Function description	Subtract 1 hour (winter time change)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR SUB1H LL_RTC_TIME_DecHour

LL_RTC_TIME_IncHour

Function name	<code>_STATIC_INLINE void LL_RTC_TIME_IncHour (RTC_TypeDef * RTCx)</code>
Function description	Add 1 hour (summer time change)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR ADD1H LL_RTC_TIME_IncHour

LL_RTC_TIME_GetSubSecond

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_TIME_GetSubSecond (RTC_TypeDef * RTCx)</code>
Function description	Get Sub second value in the synchronous prescaler counter.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Sub: second value (number between 0 and 65535)
Notes	<ul style="list-style-type: none"> • You can use both SubSeconds value and SecondFraction (PREDIV_S through LL_RTC_GetSynchPrescaler function) terms returned to convert Calendar SubSeconds value in second fraction ratio with time unit following generic formula: $\text{Seconds fraction ratio} * \text{time_unit} = [(\text{SecondFraction} - \text{SubSeconds}) / (\text{SecondFraction} + 1)] * \text{time_unit}$ <p>This conversion can be performed only if no shift operation is pending (ie. SHFP=0) when PREDIV_S >= SS.</p>
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SSR SS LL_RTC_TIME_GetSubSecond

LL_RTC_TIME_Synchronize

Function name	<code>_STATIC_INLINE void LL_RTC_TIME_Synchronize (RTC_TypeDef * RTCx, uint32_t ShiftSecond, uint32_t Fraction)</code>
Function description	Synchronize to a remote clock with a high degree of precision.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • ShiftSecond: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_SHIFT_SECOND_DELAY - LL_RTC_SHIFT_SECOND_ADVANCE • Fraction: Number of Seconds Fractions (any value from 0 to 0x7FFF)
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This operation effectively subtracts from (delays) or advance

- the clock of a fraction of a second.
 - Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
 - When REFCKON is set, firmware must not write to Shift control register.
- Reference Manual to LL API cross reference:
- SHIFTR ADD1S LL_RTC_TIME_Synchronize
 - SHIFTR SUBFS LL_RTC_TIME_Synchronize

LL_RTC_DATE_SetYear

Function name	<code>__STATIC_INLINE void LL_RTC_DATE_SetYear(RTC_TypeDef * RTCx, uint32_t Year)</code>
Function description	Set Year in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Year: Value between Min_Data=0x00 and Max_Data=0x99
Return values	None:
Notes	<ul style="list-style-type: none"> helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Year from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DR YT LL_RTC_DATE_SetYear DR YU LL_RTC_DATE_SetYear

LL_RTC_DATE_GetYear

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_DATE_GetYear(RTC_TypeDef * RTCx)</code>
Function description	Get Year in BCD format.
Parameters	RTCx: RTC Instance
Return values	Value: between Min_Data=0x00 and Max_Data=0x99
Notes	<ul style="list-style-type: none"> if shadow mode is disabled (BYPHAD=0), need to check if RSF flag is set before reading this bit helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Year from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DR YT LL_RTC_DATE_GetYear DR YU LL_RTC_DATE_GetYear

LL_RTC_DATE_SetWeekDay

Function name	<code>__STATIC_INLINE void LL_RTC_DATE_SetWeekDay(RTC_TypeDef * RTCx, uint32_t WeekDay)</code>
Function description	Set Week day.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance WeekDay: This parameter can be one of the following values: <ul style="list-style-type: none"> LL_RTC_WEEKDAY_MONDAY

- LL_RTC_WEEKDAY_TUESDAY
- LL_RTC_WEEKDAY_WEDNESDAY
- LL_RTC_WEEKDAY_THURSDAY
- LL_RTC_WEEKDAY_FRIDAY
- LL_RTC_WEEKDAY_SATURDAY
- LL_RTC_WEEKDAY_SUNDAY

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- DR WDU LL_RTC_DATE_SetWeekDay

LL_RTC_DATE_GetWeekDay

Function name **`__STATIC_INLINE uint32_t LL_RTC_DATE_GetWeekDay(RTC_TypeDef * RTCx)`**

Function description Get Week day.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:

- LL_RTC_WEEKDAY_MONDAY
- LL_RTC_WEEKDAY_TUESDAY
- LL_RTC_WEEKDAY_WEDNESDAY
- LL_RTC_WEEKDAY_THURSDAY
- LL_RTC_WEEKDAY_FRIDAY
- LL_RTC_WEEKDAY_SATURDAY
- LL_RTC_WEEKDAY_SUNDAY

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit

Reference Manual to
LL API cross
reference:

- DR WDU LL_RTC_DATE_GetWeekDay

LL_RTC_DATE_SetMonth

Function name **`__STATIC_INLINE void LL_RTC_DATE_SetMonth(RTC_TypeDef * RTCx, uint32_t Month)`**

Function description Set Month in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Month:** This parameter can be one of the following values:
 - LL_RTC_MONTH_JANUARY
 - LL_RTC_MONTH_FEBRUARY
 - LL_RTC_MONTH_MARCH
 - LL_RTC_MONTH_APRI
 - LL_RTC_MONTH_MAY
 - LL_RTC_MONTH_JUNE
 - LL_RTC_MONTH_JULY
 - LL_RTC_MONTH_AUGUST
 - LL_RTC_MONTH_SEPTEMBER
 - LL_RTC_MONTH_OCTOBER
 - LL_RTC_MONTH_NOVEMBER

	– LL_RTC_MONTH_DECEMBER
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Month from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DR MT LL_RTC_DATE_SetMonth • DR MU LL_RTC_DATE_SetMonth

LL_RTC_DATE_GetMonth

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_DATE_GetMonth(RTC_TypeDef * RTCx)</code>
Function description	Get Month in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_MONTH_JANUARY – LL_RTC_MONTH_FEBRUARY – LL_RTC_MONTH_MARCH – LL_RTC_MONTH_APRIIL – LL_RTC_MONTH_MAY – LL_RTC_MONTH_JUNE – LL_RTC_MONTH_JULY – LL_RTC_MONTH_AUGUST – LL_RTC_MONTH_SEPTEMBER – LL_RTC_MONTH_OCTOBER – LL_RTC_MONTH_NOVEMBER – LL_RTC_MONTH_DECEMBER
Notes	<ul style="list-style-type: none"> • if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit • helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Month from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DR MT LL_RTC_DATE_GetMonth • DR MU LL_RTC_DATE_GetMonth

LL_RTC_DATE_SetDay

Function name	<code>__STATIC_INLINE void LL_RTC_DATE_SetDay(RTC_TypeDef * RTCx, uint32_t Day)</code>
Function description	Set Day in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Day: Value between Min_Data=0x01 and Max_Data=0x31
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Day from binary to BCD format
Reference Manual to LL API cross	<ul style="list-style-type: none"> • DR DT LL_RTC_DATE_SetDay

-
- reference:
- DR DU LL_RTC_DATE_SetDay

LL_RTC_DATE_GetDay

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_DATE_GetDay (RTC_TypeDef * RTCx)</code>
Function description	Get Day in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x01 and Max_Data=0x31
Notes	<ul style="list-style-type: none"> if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Day from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DR DT LL_RTC_DATE_GetDay DR DU LL_RTC_DATE_GetDay

LL_RTC_DATE_Config

Function name	<code>_STATIC_INLINE void LL_RTC_DATE_Config (RTC_TypeDef * RTCx, uint32_t WeekDay, uint32_t Day, uint32_t Month, uint32_t Year)</code>
Function description	Set date (WeekDay, Day, Month and Year) in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance WeekDay: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_WEEKDAY_MONDAY - LL_RTC_WEEKDAY_TUESDAY - LL_RTC_WEEKDAY_WEDNESDAY - LL_RTC_WEEKDAY_THURSDAY - LL_RTC_WEEKDAY_FRIDAY - LL_RTC_WEEKDAY_SATURDAY - LL_RTC_WEEKDAY_SUNDAY Day: Value between Min_Data=0x01 and Max_Data=0x31 Month: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_MONTH_JANUARY - LL_RTC_MONTH_FEBRUARY - LL_RTC_MONTH_MARCH - LL_RTC_MONTH_APRI - LL_RTC_MONTH_MAY - LL_RTC_MONTH_JUNE - LL_RTC_MONTH_JULY - LL_RTC_MONTH_AUGUST - LL_RTC_MONTH_SEPTMBER - LL_RTC_MONTH_OCTOBER - LL_RTC_MONTH_NOVEMBER - LL_RTC_MONTH_DECEMBER Year: Value between Min_Data=0x00 and Max_Data=0x99 None:
Return values	

Reference Manual to LL API cross reference: DR WDU LL_RTC_DATE_Config DR MT LL_RTC_DATE_Config DR MU LL_RTC_DATE_Config DR DT LL_RTC_DATE_Config DR DU LL_RTC_DATE_Config DR YT LL_RTC_DATE_Config DR YU LL_RTC_DATE_Config

LL_RTC_DATE_Get

Function name _STATIC_INLINE uint32_t LL_RTC_DATE_Get (RTC_TypeDef * RTCx)
Function description Get date (WeekDay, Day, Month and Year) in BCD format.
Parameters <ul style="list-style-type: none"> • RTCx: RTC Instance
Return values <ul style="list-style-type: none"> • Combination: of WeekDay, Day, Month and Year (Format: 0xWWDDMMYY).
Notes <ul style="list-style-type: none"> • if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit • helper macros __LL_RTC_GET_WEEKDAY, __LL_RTC_GET_YEAR, __LL_RTC_GET_MONTH, and __LL_RTC_GET_DAY are available to get independently each parameter.
Reference Manual to LL API cross reference: DR WDU LL_RTC_DATE_Get DR MT LL_RTC_DATE_Get DR MU LL_RTC_DATE_Get DR DT LL_RTC_DATE_Get DR DU LL_RTC_DATE_Get DR YT LL_RTC_DATE_Get DR YU LL_RTC_DATE_Get

LL_RTC_ALMA_Enable

Function name _STATIC_INLINE void LL_RTC_ALMA_Enable (RTC_TypeDef * RTCx)
Function description Enable Alarm A.
Parameters <ul style="list-style-type: none"> • RTCx: RTC Instance
Return values <ul style="list-style-type: none"> • None:
Notes <ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference: CR ALRAE LL_RTC_ALMA_Enable

LL_RTC_ALMA_Disable

Function name _STATIC_INLINE void LL_RTC_ALMA_Disable (RTC_TypeDef * RTCx)
--

Function description	Disable Alarm A.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR ALRAE LL_RTC_ALMA_Disable

LL_RTC_ALMA_SetMask

Function name	<code>_STATIC_INLINE void LL_RTC_ALMA_SetMask (RTC_TypeDef * RTCx, uint32_t Mask)</code>
Function description	Specify the Alarm A masks.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Mask: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_RTC_ALMA_MASK_NONE - LL_RTC_ALMA_MASK_DATEWEEKDAY - LL_RTC_ALMA_MASK_HOURS - LL_RTC_ALMA_MASK_MINUTES - LL_RTC_ALMA_MASK_SECONDS - LL_RTC_ALMA_MASK_ALL
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR MSK4 LL_RTC_ALMA_SetMask • ALRMAR MSK3 LL_RTC_ALMA_SetMask • ALRMAR MSK2 LL_RTC_ALMA_SetMask • ALRMAR MSK1 LL_RTC_ALMA_SetMask

LL_RTC_ALMA_GetMask

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_ALMA_GetMask (RTC_TypeDef * RTCx)</code>
Function description	Get the Alarm A masks.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be can be a combination of the following values: <ul style="list-style-type: none"> - LL_RTC_ALMA_MASK_NONE - LL_RTC_ALMA_MASK_DATEWEEKDAY - LL_RTC_ALMA_MASK_HOURS - LL_RTC_ALMA_MASK_MINUTES - LL_RTC_ALMA_MASK_SECONDS - LL_RTC_ALMA_MASK_ALL
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR MSK4 LL_RTC_ALMA_GetMask • ALRMAR MSK3 LL_RTC_ALMA_GetMask • ALRMAR MSK2 LL_RTC_ALMA_GetMask • ALRMAR MSK1 LL_RTC_ALMA_GetMask

LL_RTC_ALMA_EnableWeekday

Function name	<code>_STATIC_INLINE void LL_RTC_ALMA_EnableWeekday(RTC_TypeDef * RTCx)</code>
Function description	Enable AlarmA Week day selection (DU[3:0] represents the week day).
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR WDSEL LL_RTC_ALMA_EnableWeekday

LL_RTC_ALMA_DisableWeekday

Function name	<code>_STATIC_INLINE void LL_RTC_ALMA_DisableWeekday(RTC_TypeDef * RTCx)</code>
Function description	Disable AlarmA Week day selection (DU[3:0] represents the date)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR WDSEL LL_RTC_ALMA_DisableWeekday

LL_RTC_ALMA_SetDay

Function name	<code>_STATIC_INLINE void LL_RTC_ALMA_SetDay(RTC_TypeDef * RTCx, uint32_t Day)</code>
Function description	Set ALARM A Day in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Day: Value between Min_Data=0x01 and Max_Data=0x31
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Day from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR DT LL_RTC_ALMA_SetDay • ALRMAR DU LL_RTC_ALMA_SetDay

LL_RTC_ALMA_GetDay

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_ALMA_GetDay(RTC_TypeDef * RTCx)</code>
Function description	Get ALARM A Day in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x01 and Max_Data=0x31
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BCD2BIN is available

to convert Day from BCD to Binary format

Reference Manual to
LL API cross
reference:

- ALRMAR DT LL_RTC_ALMA_GetDay
- ALRMAR DU LL_RTC_ALMA_GetDay

LL_RTC_ALMA_SetWeekDay

Function name	<code>__STATIC_INLINE void LL_RTC_ALMA_SetWeekDay((RTC_TypeDef * RTCx, uint32_t WeekDay)</code>
Function description	Set ALARM A Weekday.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • WeekDay: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_WEEKDAY_MONDAY - LL_RTC_WEEKDAY_TUESDAY - LL_RTC_WEEKDAY_WEDNESDAY - LL_RTC_WEEKDAY_THURSDAY - LL_RTC_WEEKDAY_FRIDAY - LL_RTC_WEEKDAY_SATURDAY - LL_RTC_WEEKDAY_SUNDAY
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR DU LL_RTC_ALMA_SetWeekDay

LL_RTC_ALMA_GetWeekDay

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMA_GetWeekDay((RTC_TypeDef * RTCx)</code>
Function description	Get ALARM A Weekday.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_WEEKDAY_MONDAY - LL_RTC_WEEKDAY_TUESDAY - LL_RTC_WEEKDAY_WEDNESDAY - LL_RTC_WEEKDAY_THURSDAY - LL_RTC_WEEKDAY_FRIDAY - LL_RTC_WEEKDAY_SATURDAY - LL_RTC_WEEKDAY_SUNDAY
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR DU LL_RTC_ALMA_GetWeekDay

LL_RTC_ALMA_SetTimeFormat

Function name	<code>__STATIC_INLINE void LL_RTC_ALMA_SetTimeFormat((RTC_TypeDef * RTCx, uint32_t TimeFormat)</code>
Function description	Set Alarm A time format (AM/24-hour or PM notation)

Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance TimeFormat: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_ALMA_TIME_FORMAT_AM – LL_RTC_ALMA_TIME_FORMAT_PM
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMAR PM LL_RTC_ALMA_SetTimeFormat

LL_RTC_ALMA_GetTimeFormat

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMA_GetTimeFormat(RTC_TypeDef * RTCx)</code>
Function description	Get Alarm A time format (AM or PM notation)
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_ALMA_TIME_FORMAT_AM – LL_RTC_ALMA_TIME_FORMAT_PM
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMAR PM LL_RTC_ALMA_GetTimeFormat

LL_RTC_ALMA_SetHour

Function name	<code>__STATIC_INLINE void LL_RTC_ALMA_SetHour(RTC_TypeDef * RTCx, uint32_t Hours)</code>
Function description	Set ALARM A Hours in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Hours: Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Hours from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMAR HT LL_RTC_ALMA_SetHour ALRMAR HU LL_RTC_ALMA_SetHour

LL_RTC_ALMA_GetHour

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMA_GetHour(RTC_TypeDef * RTCx)</code>
Function description	Get ALARM A Hours in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Notes	<ul style="list-style-type: none"> helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Hours from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMAR HT LL_RTC_ALMA_GetHour ALRMAR HU LL_RTC_ALMA_GetHour

LL_RTC_ALMA_SetMinute

Function name	<code>_STATIC_INLINE void LL_RTC_ALMA_SetMinute (RTC_TypeDef * RTCx, uint32_t Minutes)</code>
Function description	Set ALARM A Minutes in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Minutes: Value between Min_Data=0x00 and Max_Data=0x59
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Minutes from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMAR MNT LL_RTC_ALMA_SetMinute ALRMAR MNU LL_RTC_ALMA_SetMinute

LL_RTC_ALMA_GetMinute

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_ALMA_GetMinute (RTC_TypeDef * RTCx)</code>
Function description	Get ALARM A Minutes in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x00 and Max_Data=0x59
Notes	<ul style="list-style-type: none"> helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Minutes from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMAR MNT LL_RTC_ALMA_GetMinute ALRMAR MNU LL_RTC_ALMA_GetMinute

LL_RTC_ALMA_SetSecond

Function name	<code>_STATIC_INLINE void LL_RTC_ALMA_SetSecond (RTC_TypeDef * RTCx, uint32_t Seconds)</code>
Function description	Set ALARM A Seconds in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Seconds: Value between Min_Data=0x00 and Max_Data=0x59
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Seconds from binary to BCD format
Reference Manual to	<ul style="list-style-type: none"> ALRMAR ST LL_RTC_ALMA_SetSecond

- ALRMAR SU LL_RTC_ALMA_SetSecond
- LL API cross reference:

LL_RTC_ALMA_GetSecond

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMA_GetSecond (RTC_TypeDef * RTCx)</code>
Function description	Get ALARM A Seconds in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0x59
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Seconds from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR ST LL_RTC_ALMA_GetSecond • ALRMAR SU LL_RTC_ALMA_GetSecond

LL_RTC_ALMA_ConfigTime

Function name	<code>__STATIC_INLINE void LL_RTC_ALMA_ConfigTime (RTC_TypeDef * RTCx, uint32_t Format12_24, uint32_t Hours, uint32_t Minutes, uint32_t Seconds)</code>
Function description	Set Alarm A Time (hour, minute and second) in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Format12_24: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_ALMA_TIME_FORMAT_AM - LL_RTC_ALMA_TIME_FORMAT_PM • Hours: Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23 • Minutes: Value between Min_Data=0x00 and Max_Data=0x59 • Seconds: Value between Min_Data=0x00 and Max_Data=0x59
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMAR PM LL_RTC_ALMA_ConfigTime • ALRMAR HT LL_RTC_ALMA_ConfigTime • ALRMAR HU LL_RTC_ALMA_ConfigTime • ALRMAR MNT LL_RTC_ALMA_ConfigTime • ALRMAR MNU LL_RTC_ALMA_ConfigTime • ALRMAR ST LL_RTC_ALMA_ConfigTime • ALRMAR SU LL_RTC_ALMA_ConfigTime

LL_RTC_ALMA_GetTime

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMA_GetTime (RTC_TypeDef * RTCx)</code>
Function description	Get Alarm B Time (hour, minute and second) in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance

Return values	<ul style="list-style-type: none"> Combination: of hours, minutes and seconds.
Notes	<ul style="list-style-type: none"> helper macros <code>__LL_RTC_GET_HOUR</code>, <code>__LL_RTC_GET_MINUTE</code> and <code>__LL_RTC_GET_SECOND</code> are available to get independently each parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMAR HT LL_RTC_ALMA_GetTime ALRMAR HU LL_RTC_ALMA_GetTime ALRMAR MNT LL_RTC_ALMA_GetTime ALRMAR MNU LL_RTC_ALMA_GetTime ALRMAR ST LL_RTC_ALMA_GetTime ALRMAR SU LL_RTC_ALMA_GetTime

LL_RTC_ALMA_SetSubSecondMask

Function name	<code>__STATIC_INLINE void LL_RTC_ALMA_SetSubSecondMask (RTC_TypeDef * RTCx, uint32_t Mask)</code>
Function description	Set Alarm A Mask the most-significant bits starting at this bit.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Mask: Value between Min_Data=0x00 and Max_Data=0xF
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMASSR MASKSS LL_RTC_ALMA_SetSubSecondMask

LL_RTC_ALMA_GetSubSecondMask

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMA_GetSubSecondMask (RTC_TypeDef * RTCx)</code>
Function description	Get Alarm A Mask the most-significant bits starting at this bit.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x00 and Max_Data=0xF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> ALRMASSR MASKSS LL_RTC_ALMA_GetSubSecondMask

LL_RTC_ALMA_SetSubSecond

Function name	<code>__STATIC_INLINE void LL_RTC_ALMA_SetSubSecond (RTC_TypeDef * RTCx, uint32_t Subsecond)</code>
Function description	Set Alarm A Sub seconds value.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Subsecond: Value between Min_Data=0x00 and Max_Data=0x7FFF
Return values	<ul style="list-style-type: none"> None:
Reference Manual to	<ul style="list-style-type: none"> ALRMASSR SS LL_RTC_ALMA_SetSubSecond

LL API cross
reference:

LL_RTC_ALMA_GetSubSecond

Function name **_STATIC_INLINE uint32_t LL_RTC_ALMA_GetSubSecond
(RTC_TypeDef * RTCx)**

Function description Get Alarm A Sub seconds value.

Parameters • **RTCx:** RTC Instance

Return values • **Value:** between Min_Data=0x00 and Max_Data=0x7FFF

Reference Manual to
LL API cross
reference:

LL_RTC_ALMB_Enable

Function name **_STATIC_INLINE void LL_RTC_ALMB_Enable (RTC_TypeDef
* RTCx)**

Function description Enable Alarm B.

Parameters • **RTCx:** RTC Instance

Return values • **None:**

Notes • Bit is write-protected. LL_RTC_DisableWriteProtection
function should be called before.

Reference Manual to
LL API cross
reference:

LL_RTC_ALMB_Disable

Function name **_STATIC_INLINE void LL_RTC_ALMB_Disable
(RTC_TypeDef * RTCx)**

Function description Disable Alarm B.

Parameters • **RTCx:** RTC Instance

Return values • **None:**

Notes • Bit is write-protected. LL_RTC_DisableWriteProtection
function should be called before.

Reference Manual to
LL API cross
reference:

LL_RTC_ALMB_SetMask

Function name **_STATIC_INLINE void LL_RTC_ALMB_SetMask
(RTC_TypeDef * RTCx, uint32_t Mask)**

Function description Specify the Alarm B masks.

Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance Mask: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_RTC_ALMB_MASK_NONE - LL_RTC_ALMB_MASK_DATEWEEKDAY - LL_RTC_ALMB_MASK_HOURS - LL_RTC_ALMB_MASK_MINUTES - LL_RTC_ALMB_MASK_SECONDS - LL_RTC_ALMB_MASK_ALL
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR MSK4 LL_RTC_ALMB_SetMask • ALRMBR MSK3 LL_RTC_ALMB_SetMask • ALRMBR MSK2 LL_RTC_ALMB_SetMask • ALRMBR MSK1 LL_RTC_ALMB_SetMask

LL_RTC_ALMB_GetMask

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_ALMB_GetMask((RTC_TypeDef * RTCx))</code>
Function description	Get the Alarm B masks.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Returned: value can be can be a combination of the following values: <ul style="list-style-type: none"> - LL_RTC_ALMB_MASK_NONE - LL_RTC_ALMB_MASK_DATEWEEKDAY - LL_RTC_ALMB_MASK_HOURS - LL_RTC_ALMB_MASK_MINUTES - LL_RTC_ALMB_MASK_SECONDS - LL_RTC_ALMB_MASK_ALL
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR MSK4 LL_RTC_ALMB_GetMask • ALRMBR MSK3 LL_RTC_ALMB_GetMask • ALRMBR MSK2 LL_RTC_ALMB_GetMask • ALRMBR MSK1 LL_RTC_ALMB_GetMask

LL_RTC_ALMB_EnableWeekday

Function name	<code>_STATIC_INLINE void LL_RTC_ALMB_EnableWeekday((RTC_TypeDef * RTCx))</code>
Function description	Enable AlarmB Week day selection (DU[3:0] represents the week day).
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR WDSEL LL_RTC_ALMB_EnableWeekday

LL_RTC_ALMB_DisableWeekday

Function name	<code>__STATIC_INLINE void LL_RTC_ALMB_DisableWeekday (RTC_TypeDef * RTCx)</code>
Function description	Disable AlarmB Week day selection (DU[3:0] represents the date)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR WDSEL LL_RTC_ALMB_DisableWeekday

LL_RTC_ALMB_SetDay

Function name	<code>__STATIC_INLINE void LL_RTC_ALMB_SetDay (RTC_TypeDef * RTCx, uint32_t Day)</code>
Function description	Set ALARM B Day in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Day: Value between Min_Data=0x01 and Max_Data=0x31
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Day from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR DT LL_RTC_ALMB_SetDay • ALRMBR DU LL_RTC_ALMB_SetDay

LL_RTC_ALMB_GetDay

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMB_GetDay (RTC_TypeDef * RTCx)</code>
Function description	Get ALARM B Day in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x01 and Max_Data=0x31
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Day from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR DT LL_RTC_ALMB_GetDay • ALRMBR DU LL_RTC_ALMB_GetDay

LL_RTC_ALMB_SetWeekDay

Function name	<code>__STATIC_INLINE void LL_RTC_ALMB_SetWeekDay (RTC_TypeDef * RTCx, uint32_t WeekDay)</code>
Function description	Set ALARM B Weekday.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • WeekDay: This parameter can be one of the following values:

- LL_RTC_WEEKDAY_MONDAY
- LL_RTC_WEEKDAY_TUESDAY
- LL_RTC_WEEKDAY_WEDNESDAY
- LL_RTC_WEEKDAY_THURSDAY
- LL_RTC_WEEKDAY_FRIDAY
- LL_RTC_WEEKDAY_SATURDAY
- LL_RTC_WEEKDAY_SUNDAY

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- ALRMBR DU LL_RTC_ALMB_SetWeekDay

LL_RTC_ALMB_GetWeekDay

Function name **__STATIC_INLINE uint32_t LL_RTC_ALMB_GetWeekDay(RTC_TypeDef * RTCx)**

Function description Get ALARM B Weekday.

Parameters **RTCx:** RTC InstanceReturn values **Returned:** value can be one of the following values:

- LL_RTC_WEEKDAY_MONDAY
- LL_RTC_WEEKDAY_TUESDAY
- LL_RTC_WEEKDAY_WEDNESDAY
- LL_RTC_WEEKDAY_THURSDAY
- LL_RTC_WEEKDAY_FRIDAY
- LL_RTC_WEEKDAY_SATURDAY
- LL_RTC_WEEKDAY_SUNDAY

Reference Manual to
LL API cross
reference:

- ALRMBR DU LL_RTC_ALMB_GetWeekDay

LL_RTC_ALMB_SetTimeFormat

Function name **__STATIC_INLINE void LL_RTC_ALMB_SetTimeFormat(RTC_TypeDef * RTCx, uint32_t TimeFormat)**

Function description Set ALARM B time format (AM/24-hour or PM notation)

Parameters **RTCx:** RTC Instance**TimeFormat:** This parameter can be one of the following values:

- LL_RTC_ALMB_TIME_FORMAT_AM
- LL_RTC_ALMB_TIME_FORMAT_PM

Return values **None:**Reference Manual to
LL API cross
reference:

- ALRMBR PM LL_RTC_ALMB_SetTimeFormat

LL_RTC_ALMB_GetTimeFormat

Function name **__STATIC_INLINE uint32_t LL_RTC_ALMB_GetTimeFormat**

(RTC_TypeDef * RTCx)

Function description Get ALARM B time format (AM or PM notation)

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
 - LL_RTC_ALMB_TIME_FORMAT_AM
 - LL_RTC_ALMB_TIME_FORMAT_PM

**Reference Manual to
LL API cross
reference:**

- ALRMBR PM LL_RTC_ALMB_GetTimeFormat

LL_RTC_ALMB_SetHour

Function name **__STATIC_INLINE void LL_RTC_ALMB_SetHour
(RTC_TypeDef * RTCx, uint32_t Hours)**

Function description Set ALARM B Hours in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Hours:** Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Hours from binary to BCD format

**Reference Manual to
LL API cross
reference:**

- ALRMBR HT LL_RTC_ALMB_SetHour
- ALRMBR HU LL_RTC_ALMB_SetHour

LL_RTC_ALMB_GetHour

Function name **__STATIC_INLINE uint32_t LL_RTC_ALMB_GetHour
(RTC_TypeDef * RTCx)**

Function description Get ALARM B Hours in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Hours from BCD to Binary format

**Reference Manual to
LL API cross
reference:**

- ALRMBR HT LL_RTC_ALMB_GetHour
- ALRMBR HU LL_RTC_ALMB_GetHour

LL_RTC_ALMB_SetMinute

Function name **__STATIC_INLINE void LL_RTC_ALMB_SetMinute
(RTC_TypeDef * RTCx, uint32_t Minutes)**

Function description Set ALARM B Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance

	<ul style="list-style-type: none"> • Minutes: between Min_Data=0x00 and Max_Data=0x59
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Minutes from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR MNT LL_RTC_ALMB_SetMinute • ALRMBR MNU LL_RTC_ALMB_SetMinute

LL_RTC_ALMB_GetMinute

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMB_GetMinute(RTC_TypeDef * RTCx)</code>
Function description	Get ALARM B Minutes in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0x59
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Minutes from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR MNT LL_RTC_ALMB_GetMinute • ALRMBR MNU LL_RTC_ALMB_GetMinute

LL_RTC_ALMB_SetSecond

Function name	<code>__STATIC_INLINE void LL_RTC_ALMB_SetSecond(RTC_TypeDef * RTCx, uint32_t Seconds)</code>
Function description	Set ALARM B Seconds in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Seconds: Value between Min_Data=0x00 and Max_Data=0x59
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Seconds from binary to BCD format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR ST LL_RTC_ALMB_SetSecond • ALRMBR SU LL_RTC_ALMB_SetSecond

LL_RTC_ALMB_GetSecond

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_ALMB_GetSecond(RTC_TypeDef * RTCx)</code>
Function description	Get ALARM B Seconds in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0x59

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Seconds from BCD to Binary format

- Reference Manual to
LL API cross
reference:
- ALRMBR ST LL_RTC_ALMB_GetSecond
 - ALRMBR SU LL_RTC_ALMB_GetSecond

LL_RTC_ALMB_ConfigTime

Function name	<code>_STATIC_INLINE void LL_RTC_ALMB_ConfigTime (RTC_TypeDef * RTCx, uint32_t Format12_24, uint32_t Hours, uint32_t Minutes, uint32_t Seconds)</code>
Function description	Set Alarm B Time (hour, minute and second) in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Format12_24: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_ALMB_TIME_FORMAT_AM - LL_RTC_ALMB_TIME_FORMAT_PM • Hours: Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23 • Minutes: Value between Min_Data=0x00 and Max_Data=0x59 • Seconds: Value between Min_Data=0x00 and Max_Data=0x59
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR PM LL_RTC_ALMB_ConfigTime • ALRMBR HT LL_RTC_ALMB_ConfigTime • ALRMBR HU LL_RTC_ALMB_ConfigTime • ALRMBR MNT LL_RTC_ALMB_ConfigTime • ALRMBR MNU LL_RTC_ALMB_ConfigTime • ALRMBR ST LL_RTC_ALMB_ConfigTime • ALRMBR SU LL_RTC_ALMB_ConfigTime

LL_RTC_ALMB_GetTime

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_ALMB_GetTime (RTC_TypeDef * RTCx)</code>
Function description	Get Alarm B Time (hour, minute and second) in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Combination: of hours, minutes and seconds.
Notes	<ul style="list-style-type: none"> • helper macros <code>__LL_RTC_GET_HOUR</code>, <code>__LL_RTC_GET_MINUTE</code> and <code>__LL_RTC_GET_SECOND</code> are available to get independently each parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBR HT LL_RTC_ALMB_GetTime • ALRMBR HU LL_RTC_ALMB_GetTime • ALRMBR MNT LL_RTC_ALMB_GetTime • ALRMBR MNU LL_RTC_ALMB_GetTime • ALRMBR ST LL_RTC_ALMB_GetTime • ALRMBR SU LL_RTC_ALMB_GetTime

LL_RTC_ALMB_SetSubSecondMask

Function name	<code>_STATIC_INLINE void LL_RTC_ALMB_SetSubSecondMask(RTC_TypeDef * RTCx, uint32_t Mask)</code>
Function description	Set Alarm B Mask the most-significant bits starting at this bit.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Mask: Value between Min_Data=0x00 and Max_Data=0xF
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBSSR MASKSS LL_RTC_ALMB_SetSubSecondMask

LL_RTC_ALMB_GetSubSecondMask

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_ALMB_GetSubSecondMask(RTC_TypeDef * RTCx)</code>
Function description	Get Alarm B Mask the most-significant bits starting at this bit.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0xF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBSSR MASKSS LL_RTC_ALMB_GetSubSecondMask

LL_RTC_ALMB_SetSubSecond

Function name	<code>_STATIC_INLINE void LL_RTC_ALMB_SetSubSecond(RTC_TypeDef * RTCx, uint32_t Subsecond)</code>
Function description	Set Alarm B Sub seconds value.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Subsecond: Value between Min_Data=0x00 and Max_Data=0x7FFF
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ALRMBSSR SS LL_RTC_ALMB_SetSubSecond

LL_RTC_ALMB_GetSubSecond

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_ALMB_GetSubSecond(RTC_TypeDef * RTCx)</code>
Function description	Get Alarm B Sub seconds value.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0x7FFF

- Reference Manual to
LL API cross
reference:
- ALRMBSSR SS LL_RTC_ALMB_GetSubSecond

LL_RTC_TS_Enable

Function name	_STATIC_INLINE void LL_RTC_TS_Enable (RTC_TypeDef * RTCx)
Function description	Enable Timestamp.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TSE LL_RTC_TS_Enable

LL_RTC_TS_Disable

Function name	_STATIC_INLINE void LL_RTC_TS_Disable (RTC_TypeDef * RTCx)
Function description	Disable Timestamp.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TSE LL_RTC_TS_Disable

LL_RTC_TS_SetActiveEdge

Function name	_STATIC_INLINE void LL_RTC_TS_SetActiveEdge (RTC_TypeDef * RTCx, uint32_t Edge)
Function description	Set Time-stamp event active edge.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Edge: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TIMESTAMP_EDGE_RISING - LL_RTC_TIMESTAMP_EDGE_FALLING
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. • TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting
Reference Manual to LL API cross	<ul style="list-style-type: none"> • CR TSEDGE LL_RTC_TS_SetActiveEdge

reference:

LL_RTC_TS_GetActiveEdge

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_TS_GetActiveEdge(RTC_TypeDef * RTCx)</code>
Function description	Get Time-stamp event active edge.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TIMESTAMP_EDGE_RISING - LL_RTC_TIMESTAMP_EDGE_FALLING
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TSEDGE LL_RTC_TS_GetActiveEdge

LL_RTC_TS_GetTimeFormat

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_TS_GetTimeFormat(RTC_TypeDef * RTCx)</code>
Function description	Get Timestamp AM/PM notation (AM or 24-hour format)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TS_TIME_FORMAT_AM - LL_RTC_TS_TIME_FORMAT_PM
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TSTR PM LL_RTC_TS_GetTimeFormat

LL_RTC_TS_GetHour

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_TS_GetHour(RTC_TypeDef * RTCx)</code>
Function description	Get Timestamp Hours in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Hours from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TSTR HT LL_RTC_TS_GetHour • TSTR HU LL_RTC_TS_GetHour

LL_RTC_TS_GetMinute

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TS_GetMinute(RTC_TypeDef * RTCx)</code>
Function description	Get Timestamp Minutes in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0x59
Notes	<ul style="list-style-type: none"> • helper macro <code>__LL_RTC_CONVERT_BCD2BIN</code> is available to convert Minutes from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TSTR MNT LL_RTC_TS_GetMinute • TSTR MNU LL_RTC_TS_GetMinute

LL_RTC_TS_GetSecond

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TS_GetSecond(RTC_TypeDef * RTCx)</code>
Function description	Get Timestamp Seconds in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0x59
Notes	<ul style="list-style-type: none"> • helper macro <code>__LL_RTC_CONVERT_BCD2BIN</code> is available to convert Seconds from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TSTR ST LL_RTC_TS_GetSecond • TSTR SU LL_RTC_TS_GetSecond

LL_RTC_TS_GetTime

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TS_GetTime(RTC_TypeDef * RTCx)</code>
Function description	Get Timestamp time (hour, minute and second) in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Combination: of hours, minutes and seconds.
Notes	<ul style="list-style-type: none"> • helper macros <code>__LL_RTC_GET_HOUR</code>, <code>__LL_RTC_GET_MINUTE</code> and <code>__LL_RTC_GET_SECOND</code> are available to get independently each parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TSTR HT LL_RTC_TS_GetTime • TSTR HU LL_RTC_TS_GetTime • TSTR MNT LL_RTC_TS_GetTime • TSTR MNU LL_RTC_TS_GetTime • TSTR ST LL_RTC_TS_GetTime • TSTR SU LL_RTC_TS_GetTime

LL_RTC_TS_GetWeekDay

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TS_GetWeekDay</code>
---------------	--

(RTC_TypeDef * RTCx)

Function description	Get Timestamp Week day.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_WEEKDAY_MONDAY - LL_RTC_WEEKDAY_TUESDAY - LL_RTC_WEEKDAY_WEDNESDAY - LL_RTC_WEEKDAY_THURSDAY - LL_RTC_WEEKDAY_FRIDAY - LL_RTC_WEEKDAY_SATURDAY - LL_RTC_WEEKDAY_SUNDAY
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TSDR WDU LL_RTC_TS_GetWeekDay

LL_RTC_TS_GetMonth

Function name	__STATIC_INLINE uint32_t LL_RTC_TS_GetMonth (RTC_TypeDef * RTCx)
Function description	Get Timestamp Month in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_MONTH_JANUARY - LL_RTC_MONTH_FEBRUARY - LL_RTC_MONTH_MARCH - LL_RTC_MONTH_APRIIL - LL_RTC_MONTH_MAY - LL_RTC_MONTH_JUNE - LL_RTC_MONTH_JULY - LL_RTC_MONTH_AUGUST - LL_RTC_MONTH_SEPTEMBER - LL_RTC_MONTH_OCTOBER - LL_RTC_MONTH_NOVEMBER - LL_RTC_MONTH_DECEMBER
Notes	<ul style="list-style-type: none"> • helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Month from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TSDR MT LL_RTC_TS_GetMonth • TSDR MU LL_RTC_TS_GetMonth

LL_RTC_TS_GetDay

Function name	__STATIC_INLINE uint32_t LL_RTC_TS_GetDay (RTC_TypeDef * RTCx)
Function description	Get Timestamp Day in BCD format.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x01 and Max_Data=0x31

Notes	<ul style="list-style-type: none"> helper macro <code>__LL_RTC_CONVERT_BCD2BIN</code> is available to convert Day from BCD to Binary format
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> TSDR DT LL_RTC_TS_GetDay TSDR DU LL_RTC_TS_GetDay

LL_RTC_TS_GetDate

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TS_GetDate (RTC_TypeDef * RTCx)</code>
Function description	Get Timestamp date (WeekDay, Day and Month) in BCD format.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Combination: of Weekday, Day and Month
Notes	<ul style="list-style-type: none"> helper macros <code>__LL_RTC_GET_WEEKDAY</code>, <code>__LL_RTC_GET_MONTH</code>, and <code>__LL_RTC_GET_DAY</code> are available to get independently each parameter.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> TSDR WDU LL_RTC_TS_GetDate TSDR MT LL_RTC_TS_GetDate TSDR MU LL_RTC_TS_GetDate TSDR DT LL_RTC_TS_GetDate TSDR DU LL_RTC_TS_GetDate

LL_RTC_TS_GetSubSecond

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TS_GetSubSecond (RTC_TypeDef * RTCx)</code>
Function description	Get time-stamp sub second value.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Value: between Min_Data=0x00 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- TSSSR SS LL_RTC_TS_GetSubSecond

LL_RTC_TS_EnableOnTamper

Function name	<code>__STATIC_INLINE void LL_RTC_TS_EnableOnTamper (RTC_TypeDef * RTCx)</code>
Function description	Activate timestamp on tamper detection event.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> None:

Reference Manual to LL API cross reference:

- TAFCR TAMPTS LL_RTC_TS_EnableOnTamper

LL_RTC_TS_DisableOnTamper

Function name	<code>__STATIC_INLINE void LL_RTC_TS_DisableOnTamper(RTC_TypeDef * RTCx)</code>
Function description	Disable timestamp on tamper detection event.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMPTS LL_RTC_TS_DisableOnTamper

LL_RTC_TAMPER_Enable

Function name	<code>__STATIC_INLINE void LL_RTC_TAMPER_Enable(RTC_TypeDef * RTCx, uint32_t Tamper)</code>
Function description	Enable RTC_TAMPx input detection.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Tamper: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RTC_TAMPER_1 - LL_RTC_TAMPER_2 (*) - LL_RTC_TAMPER_3 (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMP1E LL_RTC_TAMPER_Enable • TAFCR TAMP2E LL_RTC_TAMPER_Enable • TAFCR TAMP3E LL_RTC_TAMPER_Enable

LL_RTC_TAMPER_Disable

Function name	<code>__STATIC_INLINE void LL_RTC_TAMPER_Disable(RTC_TypeDef * RTCx, uint32_t Tamper)</code>
Function description	Clear RTC_TAMPx input detection.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Tamper: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RTC_TAMPER_1 - LL_RTC_TAMPER_2 (*) - LL_RTC_TAMPER_3 (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMP1E LL_RTC_TAMPER_Disable • TAFCR TAMP2E LL_RTC_TAMPER_Disable • TAFCR TAMP3E LL_RTC_TAMPER_Disable

LL_RTC_TAMPER_DisablePullUp

Function name	<code>__STATIC_INLINE void LL_RTC_TAMPER_DisablePullUp(RTC_TypeDef * RTCx)</code>
---------------	---

Function description	Disable RTC_TAMPx pull-up disable (Disable precharge of RTC_TAMPx pins)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	• TAFCR TAMPPUDIS LL_RTC_TAMPER_DisablePullUp

LL_RTC_TAMPER_EnablePullUp

Function name	<code>_STATIC_INLINE void LL_RTC_TAMPER_EnablePullUp (RTC_TypeDef * RTCx)</code>
Function description	Enable RTC_TAMPx pull-up disable (Precharge RTC_TAMPx pins before sampling)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	• TAFCR TAMPPUDIS LL_RTC_TAMPER_EnablePullUp

LL_RTC_TAMPER_SetPrecharge

Function name	<code>_STATIC_INLINE void LL_RTC_TAMPER_SetPrecharge (RTC_TypeDef * RTCx, uint32_t Duration)</code>
Function description	Set RTC_TAMPx precharge duration.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Duration: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TAMPER_DURATION_1RTCCLK - LL_RTC_TAMPER_DURATION_2RTCCLK - LL_RTC_TAMPER_DURATION_4RTCCLK - LL_RTC_TAMPER_DURATION_8RTCCLK
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	• TAFCR TAMPPRCH LL_RTC_TAMPER_SetPrecharge

LL_RTC_TAMPER_GetPrecharge

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_TAMPER_GetPrecharge (RTC_TypeDef * RTCx)</code>
Function description	Get RTC_TAMPx precharge duration.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TAMPER_DURATION_1RTCCLK - LL_RTC_TAMPER_DURATION_2RTCCLK - LL_RTC_TAMPER_DURATION_4RTCCLK

- Reference Manual to
LL API cross
reference:
- LL_RTC_TAMPER_DURATION_8RTCCLK
 - TAFCR TAMPPRCH LL_RTC_TAMPER_GetPrecharge

LL_RTC_TAMPER_SetFilterCount

Function name	<code>_STATIC_INLINE void LL_RTC_TAMPER_SetFilterCount (RTC_TypeDef * RTCx, uint32_t FilterCount)</code>
Function description	Set RTC_TAMPx filter count.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • FilterCount: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TAMPER_FILTER_DISABLE - LL_RTC_TAMPER_FILTER_2SAMPLE - LL_RTC_TAMPER_FILTER_4SAMPLE - LL_RTC_TAMPER_FILTER_8SAMPLE
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMPFLT LL_RTC_TAMPER_SetFilterCount

LL_RTC_TAMPER_GetFilterCount

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_TAMPER_GetFilterCount (RTC_TypeDef * RTCx)</code>
Function description	Get RTC_TAMPx filter count.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TAMPER_FILTER_DISABLE - LL_RTC_TAMPER_FILTER_2SAMPLE - LL_RTC_TAMPER_FILTER_4SAMPLE - LL_RTC_TAMPER_FILTER_8SAMPLE
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMPFLT LL_RTC_TAMPER_GetFilterCount

LL_RTC_TAMPER_SetSamplingFreq

Function name	<code>_STATIC_INLINE void LL_RTC_TAMPER_SetSamplingFreq (RTC_TypeDef * RTCx, uint32_t SamplingFreq)</code>
Function description	Set Tamper sampling frequency.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • SamplingFreq: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TAMPER_SAMPLFREQDIV_32768 - LL_RTC_TAMPER_SAMPLFREQDIV_16384 - LL_RTC_TAMPER_SAMPLFREQDIV_8192

	<ul style="list-style-type: none"> - LL_RTC_TAMPER_SAMPLFREQDIV_4096 - LL_RTC_TAMPER_SAMPLFREQDIV_2048 - LL_RTC_TAMPER_SAMPLFREQDIV_1024 - LL_RTC_TAMPER_SAMPLFREQDIV_512 - LL_RTC_TAMPER_SAMPLFREQDIV_256
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMPFREQ LL_RTC_TAMPER_SetSamplingFreq

LL_RTC_TAMPER_GetSamplingFreq

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_TAMPER_GetSamplingFreq (RTC_TypeDef * RTCx)</code>
Function description	Get Tamper sampling frequency.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_TAMPER_SAMPLFREQDIV_32768 - LL_RTC_TAMPER_SAMPLFREQDIV_16384 - LL_RTC_TAMPER_SAMPLFREQDIV_8192 - LL_RTC_TAMPER_SAMPLFREQDIV_4096 - LL_RTC_TAMPER_SAMPLFREQDIV_2048 - LL_RTC_TAMPER_SAMPLFREQDIV_1024 - LL_RTC_TAMPER_SAMPLFREQDIV_512 - LL_RTC_TAMPER_SAMPLFREQDIV_256
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMPFREQ LL_RTC_TAMPER_GetSamplingFreq

LL_RTC_TAMPER_EnableActiveLevel

Function name	<code>__STATIC_INLINE void LL_RTC_TAMPER_EnableActiveLevel (RTC_TypeDef * RTCx, uint32_t Tamper)</code>
Function description	Enable Active level for Tamper input.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Tamper: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RTC_TAMPER_ACTIVELEVEL_TAMP1 - LL_RTC_TAMPER_ACTIVELEVEL_TAMP2 (*) - LL_RTC_TAMPER_ACTIVELEVEL_TAMP3 (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMP1TRG LL_RTC_TAMPER_EnableActiveLevel • TAFCR TAMP2TRG LL_RTC_TAMPER_EnableActiveLevel • TAFCR TAMP3TRG LL_RTC_TAMPER_EnableActiveLevel

LL_RTC_TAMPER_DisableActiveLevel

Function name	<code>__STATIC_INLINE void LL_RTC_TAMPER_DisableActiveLevel</code>
---------------	--

(RTC_TypeDef * RTCx, uint32_t Tamper)

Function description	Disable Active level for Tamper input.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Tamper: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RTC_TAMPER_ACTIVELEVEL_TAMP1 - LL_RTC_TAMPER_ACTIVELEVEL_TAMP2 (*) - LL_RTC_TAMPER_ACTIVELEVEL_TAMP3 (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMP1TRG LL_RTC_TAMPER_DisableActiveLevel • TAFCR TAMP2TRG LL_RTC_TAMPER_DisableActiveLevel • TAFCR TAMP3TRG LL_RTC_TAMPER_DisableActiveLevel

LL_RTC_WAKEUP_Enable

Function name	__STATIC_INLINE void LL_RTC_WAKEUP_Enable (RTC_TypeDef * RTCx)
Function description	Enable Wakeup timer.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR WUTE LL_RTC_WAKEUP_Enable

LL_RTC_WAKEUP_Disable

Function name	__STATIC_INLINE void LL_RTC_WAKEUP_Disable (RTC_TypeDef * RTCx)
Function description	Disable Wakeup timer.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR WUTE LL_RTC_WAKEUP_Disable

LL_RTC_WAKEUP_IsEnabled

Function name	__STATIC_INLINE uint32_t LL_RTC_WAKEUP_IsEnabled (RTC_TypeDef * RTCx)
Function description	Check if Wakeup timer is enabled or not.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance

Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR WUTE LL_RTC_WAKEUP_IsEnabled

LL_RTC_WAKEUP_SetClock

Function name	<code>__STATIC_INLINE void LL_RTC_WAKEUP_SetClock (RTC_TypeDef * RTCx, uint32_t WakeupClock)</code>
Function description	Select Wakeup clock.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance WakeupClock: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_WAKEUPCLOCK_DIV_16 – LL_RTC_WAKEUPCLOCK_DIV_8 – LL_RTC_WAKEUPCLOCK_DIV_4 – LL_RTC_WAKEUPCLOCK_DIV_2 – LL_RTC_WAKEUPCLOCK_CKSPRE – LL_RTC_WAKEUPCLOCK_CKSPRE_WUT
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. Bit can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR WUCKSEL LL_RTC_WAKEUP_SetClock

LL_RTC_WAKEUP_GetClock

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_WAKEUP_GetClock (RTC_TypeDef * RTCx)</code>
Function description	Get Wakeup clock.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_WAKEUPCLOCK_DIV_16 – LL_RTC_WAKEUPCLOCK_DIV_8 – LL_RTC_WAKEUPCLOCK_DIV_4 – LL_RTC_WAKEUPCLOCK_DIV_2 – LL_RTC_WAKEUPCLOCK_CKSPRE – LL_RTC_WAKEUPCLOCK_CKSPRE_WUT
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR WUCKSEL LL_RTC_WAKEUP_GetClock

LL_RTC_WAKEUP_SetAutoReload

Function name	<code>__STATIC_INLINE void LL_RTC_WAKEUP_SetAutoReload</code>
---------------	---

(RTC_TypeDef * RTCx, uint32_t Value)

Function description	Set Wakeup auto-reload value.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Value: Value between Min_Data=0x00 and Max_Data=0xFFFF
Return values	• None:
Notes	<ul style="list-style-type: none"> • Bit can be written only when WUTWF is set to 1 in RTC_ISR
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • WUTR WUT LL_RTC_WAKEUP_SetAutoReload

LL_RTC_WAKEUP_SetAutoReload

Function name	_STATIC_INLINE uint32_t LL_RTC_WAKEUP_SetAutoReload (RTC_TypeDef * RTCx)
Function description	Get Wakeup auto-reload value.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0xFFFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • WUTR WUT LL_RTC_WAKEUP_SetAutoReload

LL_RTC_BAK_SetRegister

Function name	_STATIC_INLINE void LL_RTC_BAK_SetRegister (RTC_TypeDef * RTCx, uint32_t BackupRegister, uint32_t Data)
Function description	Writes a data in a specified RTC Backup data register.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • BackupRegister: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RTC_BKP_DR0 - LL_RTC_BKP_DR1 - LL_RTC_BKP_DR2 - LL_RTC_BKP_DR3 - LL_RTC_BKP_DR4 - LL_RTC_BKP_DR5 (*) - LL_RTC_BKP_DR6 (*) - LL_RTC_BKP_DR7 (*) - LL_RTC_BKP_DR8 (*) - LL_RTC_BKP_DR9 (*) - LL_RTC_BKP_DR10 (*) - LL_RTC_BKP_DR11 (*) - LL_RTC_BKP_DR12 (*) - LL_RTC_BKP_DR13 (*) - LL_RTC_BKP_DR14 (*) - LL_RTC_BKP_DR15 (*) - LL_RTC_BKP_DR16 (*)

- LL_RTC_BKP_DR17 (*)
- LL_RTC_BKP_DR18 (*)
- LL_RTC_BKP_DR19 (*)
- LL_RTC_BKP_DR20 (*)
- LL_RTC_BKP_DR21 (*)
- LL_RTC_BKP_DR22 (*)
- LL_RTC_BKP_DR23 (*)
- LL_RTC_BKP_DR24 (*)
- LL_RTC_BKP_DR25 (*)
- LL_RTC_BKP_DR26 (*)
- LL_RTC_BKP_DR27 (*)
- LL_RTC_BKP_DR28 (*)
- LL_RTC_BKP_DR29 (*)
- LL_RTC_BKP_DR30 (*)
- LL_RTC_BKP_DR31 (*)
- **Data:** Value between Min_Data=0x00 and Max_Data=0xFFFFFFFF

Return values

Reference Manual to
LL API cross
reference:

- **None:**

- BKPxR BKP LL_RTC_BAK_SetRegister

LL_RTC_BAK_GetRegister

Function name **STATIC_INLINE uint32_t LL_RTC_BAK_GetRegister(RTC_TypeDef * RTCx, uint32_t BackupRegister)**

Function description Reads data from the specified RTC Backup data Register.

- | | |
|------------|--|
| Parameters | <ul style="list-style-type: none"> • RTCx: RTC Instance • BackupRegister: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RTC_BKP_DR0 - LL_RTC_BKP_DR1 - LL_RTC_BKP_DR2 - LL_RTC_BKP_DR3 - LL_RTC_BKP_DR4 - LL_RTC_BKP_DR5 (*) - LL_RTC_BKP_DR6 (*) - LL_RTC_BKP_DR7 (*) - LL_RTC_BKP_DR8 (*) - LL_RTC_BKP_DR9 (*) - LL_RTC_BKP_DR10 (*) - LL_RTC_BKP_DR11 (*) - LL_RTC_BKP_DR12 (*) - LL_RTC_BKP_DR13 (*) - LL_RTC_BKP_DR14 (*) - LL_RTC_BKP_DR15 (*) - LL_RTC_BKP_DR16 (*) - LL_RTC_BKP_DR17 (*) - LL_RTC_BKP_DR18 (*) - LL_RTC_BKP_DR19 (*) - LL_RTC_BKP_DR20 (*) |
|------------|--|

- LL_RTC_BKP_DR21 (*)
- LL_RTC_BKP_DR22 (*)
- LL_RTC_BKP_DR23 (*)
- LL_RTC_BKP_DR24 (*)
- LL_RTC_BKP_DR25 (*)
- LL_RTC_BKP_DR26 (*)
- LL_RTC_BKP_DR27 (*)
- LL_RTC_BKP_DR28 (*)
- LL_RTC_BKP_DR29 (*)
- LL_RTC_BKP_DR30 (*)
- LL_RTC_BKP_DR31 (*)

Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data=0xFFFFFFFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • BKPxR BKP LL_RTC_BAK_GetRegister

LL_RTC_CAL_SetOutputFreq

Function name	<code>__STATIC_INLINE void LL_RTC_CAL_SetOutputFreq(RTC_TypeDef * RTCx, uint32_t Frequency)</code>
Function description	Set Calibration output frequency (1 Hz or 512 Hz)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Frequency: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RTC_CALIB_OUTPUT_NONE - LL_RTC_CALIB_OUTPUT_1HZ (*) - LL_RTC_CALIB_OUTPUT_512HZ
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bits are write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR COE LL_RTC_CAL_SetOutputFreq • CR COSEL LL_RTC_CAL_SetOutputFreq

LL_RTC_CAL_GetOutputFreq

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_CAL_GetOutputFreq(RTC_TypeDef * RTCx)</code>
Function description	Get Calibration output frequency (1 Hz or 512 Hz)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RTC_CALIB_OUTPUT_NONE - LL_RTC_CALIB_OUTPUT_1HZ (*) - LL_RTC_CALIB_OUTPUT_512HZ
Reference Manual to LL API cross	<ul style="list-style-type: none"> • CR COE LL_RTC_CAL_GetOutputFreq

- reference:
- CR COSEL LL_RTC_CAL_GetOutputFreq

LL_RTC_CAL_EnableCoarseDigital

Function name	<code>_STATIC_INLINE void LL_RTC_CAL_EnableCoarseDigital(RTC_TypeDef * RTCx)</code>
Function description	Enable Coarse digital calibration.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. • It can be written in initialization mode only (LL_RTC_EnableInitMode function)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR DCE LL_RTC_CAL_EnableCoarseDigital

LL_RTC_CAL_DisableCoarseDigital

Function name	<code>_STATIC_INLINE void LL_RTC_CAL_DisableCoarseDigital(RTC_TypeDef * RTCx)</code>
Function description	Disable Coarse digital calibration.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. • It can be written in initialization mode only (LL_RTC_EnableInitMode function)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR DCE LL_RTC_CAL_DisableCoarseDigital

LL_RTC_CAL_ConfigCoarseDigital

Function name	<code>_STATIC_INLINE void LL_RTC_CAL_ConfigCoarseDigital(RTC_TypeDef * RTCx, uint32_t Sign, uint32_t Value)</code>
Function description	Set the coarse digital calibration.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Sign: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_CALIB_SIGN_POSITIVE - LL_RTC_CALIB_SIGN_NEGATIVE • Value: value of coarse calibration expressed in ppm (coded on 5 bits)
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. • It can be written in initialization mode only

- (LL_RTC_EnableInitMode function)
- This Calibration value should be between 0 and 63 when using negative sign with a 2-ppm step.
 - This Calibration value should be between 0 and 126 when using positive sign with a 4-ppm step.
- Reference Manual to LL API cross reference:
- CALIBR DCS LL_RTC_CAL_ConfigCoarseDigital
 - CALIBR DC LL_RTC_CAL_ConfigCoarseDigital

LL_RTC_CAL_GetCoarseDigitalValue

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_CAL_GetCoarseDigitalValue (RTC_TypeDef * RTCx)</code>
Function description	Get the coarse digital calibration value.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • value: of coarse calibration expressed in ppm (coded on 5 bits)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CALIBR DC LL_RTC_CAL_GetCoarseDigitalValue

LL_RTC_CAL_GetCoarseDigitalSign

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_CAL_GetCoarseDigitalSign (RTC_TypeDef * RTCx)</code>
Function description	Get the coarse digital calibration sign.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_CALIB_SIGN_POSITIVE - LL_RTC_CALIB_SIGN_NEGATIVE
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CALIBR DCS LL_RTC_CAL_GetCoarseDigitalSign

LL_RTC_CAL_SetPulse

Function name	<code>__STATIC_INLINE void LL_RTC_CAL_SetPulse (RTC_TypeDef * RTCx, uint32_t Pulse)</code>
Function description	Insert or not One RTCCCLK pulse every 2^{exp11} pulses (frequency increased by 488.5 ppm)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Pulse: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_CALIB_INSERTPULSE_NONE - LL_RTC_CALIB_INSERTPULSE_SET
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

- Bit can be written only when RECALPF is set to 0 in RTC_ISR
 - CALR CALP LL_RTC_CAL_SetPulse
- Reference Manual to
LL API cross
reference:

LL_RTC_CAL_IsPulseInserted

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_CAL_IsPulseInserted(RTC_TypeDef * RTCx)</code>
Function description	Check if one RTCCLK has been inserted or not every 2 ^{exp11} pulses (frequency increased by 488.5 ppm)
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CALR CALP LL_RTC_CAL_IsPulseInserted

LL_RTC_CAL_SetPeriod

Function name	<code>_STATIC_INLINE void LL_RTC_CAL_SetPeriod(RTC_TypeDef * RTCx, uint32_t Period)</code>
Function description	Set the calibration cycle period.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • Period: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_CALIB_PERIOD_32SEC - LL_RTC_CALIB_PERIOD_16SEC - LL_RTC_CALIB_PERIOD_8SEC
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. • Bit can be written only when RECALPF is set to 0 in RTC_ISR
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CALR CALW8 LL_RTC_CAL_SetPeriod • CALR CALW16 LL_RTC_CAL_SetPeriod

LL_RTC_CAL_GetPeriod

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_CAL_GetPeriod(RTC_TypeDef * RTCx)</code>
Function description	Get the calibration cycle period.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_CALIB_PERIOD_32SEC - LL_RTC_CALIB_PERIOD_16SEC - LL_RTC_CALIB_PERIOD_8SEC

- Reference Manual to
LL API cross
reference:
- CALR CALW8 LL_RTC_CAL_GetPeriod
 - CALR CALW16 LL_RTC_CAL_GetPeriod

LL_RTC_CAL_SetMinus

Function name	<code>_STATIC_INLINE void LL_RTC_CAL_SetMinus (RTC_TypeDef * RTCx, uint32_t CalibMinus)</code>
Function description	Set Calibration minus.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • CalibMinus: Value between Min_Data=0x00 and Max_Data=0x1FF
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before. • Bit can be written only when RECALPF is set to 0 in RTC_ISR
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CALR CALM LL_RTC_CAL_SetMinus

LL_RTC_CAL_GetMinus

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_CAL_GetMinus (RTC_TypeDef * RTCx)</code>
Function description	Get Calibration minus.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • Value: between Min_Data=0x00 and Max_Data= 0x1FF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CALR CALM LL_RTC_CAL_SetMinus

LL_RTC_IsActiveFlag_RECALP

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_RECALP (RTC_TypeDef * RTCx)</code>
Function description	Get Recalibration pending Flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR RECALPF LL_RTC_IsActiveFlag_RECALP

LL_RTC_IsActiveFlag_TAMP3

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TAMP3 (RTC_TypeDef * RTCx)</code>
---------------	--

Function description	Get RTC_TAMP3 detection flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TAMP3F LL_RTC_IsActiveFlag_TAMP3

LL_RTC_IsActiveFlag_TAMP2

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TAMP2(RTC_TypeDef * RTCx)</code>
Function description	Get RTC_TAMP2 detection flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TAMP2F LL_RTC_IsActiveFlag_TAMP2

LL_RTC_IsActiveFlag_TAMP1

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TAMP1(RTC_TypeDef * RTCx)</code>
Function description	Get RTC_TAMP1 detection flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TAMP1F LL_RTC_IsActiveFlag_TAMP1

LL_RTC_IsActiveFlag_TSOV

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TSOV(RTC_TypeDef * RTCx)</code>
Function description	Get Time-stamp overflow flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TSOVF LL_RTC_IsActiveFlag_TSOV

LL_RTC_IsActiveFlag_TS

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TS(RTC_TypeDef * RTCx)</code>
Function description	Get Time-stamp flag.

Parameters	<ul style="list-style-type: none">RTCx: RTC Instance
Return values	<ul style="list-style-type: none">State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">ISR TSF LL_RTC_IsActiveFlag_TS

LL_RTC_IsActiveFlag_WUT

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_WUT (RTC_TypeDef * RTCx)</code>
Function description	Get Wakeup timer flag.
Parameters	<ul style="list-style-type: none">RTCx: RTC Instance
Return values	<ul style="list-style-type: none">State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">ISR WUTF LL_RTC_IsActiveFlag_WUT

LL_RTC_IsActiveFlag_ALRB

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALRB (RTC_TypeDef * RTCx)</code>
Function description	Get Alarm B flag.
Parameters	<ul style="list-style-type: none">RTCx: RTC Instance
Return values	<ul style="list-style-type: none">State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">ISR ALRBF LL_RTC_IsActiveFlag_ALRB

LL_RTC_IsActiveFlag_ALRA

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALRA (RTC_TypeDef * RTCx)</code>
Function description	Get Alarm A flag.
Parameters	<ul style="list-style-type: none">RTCx: RTC Instance
Return values	<ul style="list-style-type: none">State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">ISR ALRAF LL_RTC_IsActiveFlag_ALRA

LL_RTC_ClearFlag_TAMP3

Function name	<code>__STATIC_INLINE void LL_RTC_ClearFlag_TAMP3 (RTC_TypeDef * RTCx)</code>
Function description	Clear RTC_TAMP3 detection flag.
Parameters	<ul style="list-style-type: none">RTCx: RTC Instance

Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TAMP3F LL_RTC_ClearFlag_TAMP3

LL_RTC_ClearFlag_TAMP2

Function name	<code>__STATIC_INLINE void LL_RTC_ClearFlag_TAMP2 (RTC_TypeDef * RTCx)</code>
Function description	Clear RTC_TAMP2 detection flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TAMP2F LL_RTC_ClearFlag_TAMP2

LL_RTC_ClearFlag_TAMP1

Function name	<code>__STATIC_INLINE void LL_RTC_ClearFlag_TAMP1 (RTC_TypeDef * RTCx)</code>
Function description	Clear RTC_TAMP1 detection flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TAMP1F LL_RTC_ClearFlag_TAMP1

LL_RTC_ClearFlag_TSOV

Function name	<code>__STATIC_INLINE void LL_RTC_ClearFlag_TSOV (RTC_TypeDef * RTCx)</code>
Function description	Clear Time-stamp overflow flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR TSOVF LL_RTC_ClearFlag_TSOV

LL_RTC_ClearFlag_TS

Function name	<code>__STATIC_INLINE void LL_RTC_ClearFlag_TS (RTC_TypeDef * RTCx)</code>
Function description	Clear Time-stamp flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:

- Reference Manual to
LL API cross
reference:
- ISR TSF LL_RTC_ClearFlag_TS

LL_RTC_ClearFlag_WUT

Function name	<code>_STATIC_INLINE void LL_RTC_ClearFlag_WUT (RTC_TypeDef * RTCx)</code>
Function description	Clear Wakeup timer flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:

Reference Manual to
LL API cross
reference:

- ISR WUTF LL_RTC_ClearFlag_WUT

LL_RTC_ClearFlag_ALRB

Function name	<code>_STATIC_INLINE void LL_RTC_ClearFlag_ALRB (RTC_TypeDef * RTCx)</code>
Function description	Clear Alarm B flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:

Reference Manual to
LL API cross
reference:

- ISR ALRBF LL_RTC_ClearFlag_ALRB

LL_RTC_ClearFlag_ALRA

Function name	<code>_STATIC_INLINE void LL_RTC_ClearFlag_ALRA (RTC_TypeDef * RTCx)</code>
Function description	Clear Alarm A flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:

Reference Manual to
LL API cross
reference:

- ISR ALRAF LL_RTC_ClearFlag_ALRA

LL_RTC_IsActiveFlag_INIT

Function name	<code>_STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_INIT (RTC_TypeDef * RTCx)</code>
Function description	Get Initialization flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).

Reference Manual to
LL API cross

- ISR INITF LL_RTC_IsActiveFlag_INIT

reference:

LL_RTC_IsActiveFlag_RS

Function name **__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_RS
(RTC_TypeDef * RTCx)**

Function description Get Registers synchronization flag.

Parameters • **RTCx:** RTC Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
reference:

- ISR RSF LL_RTC_IsActiveFlag_RS

LL_RTC_ClearFlag_RS

Function name **__STATIC_INLINE void LL_RTC_ClearFlag_RS (RTC_TypeDef
* RTCx)**

Function description Clear Registers synchronization flag.

Parameters • **RTCx:** RTC Instance

Return values • **None:**

Reference Manual to
LL API cross
reference:
reference:

- ISR RSF LL_RTC_ClearFlag_RS

LL_RTC_IsActiveFlag_INITS

Function name **__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_INITS
(RTC_TypeDef * RTCx)**

Function description Get Initialization status flag.

Parameters • **RTCx:** RTC Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
reference:

- ISR INITS LL_RTC_IsActiveFlag_INITS

LL_RTC_IsActiveFlag_SHP

Function name **__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_SHP
(RTC_TypeDef * RTCx)**

Function description Get Shift operation pending flag.

Parameters • **RTCx:** RTC Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to
LL API cross
reference:
reference:

- ISR SHPF LL_RTC_IsActiveFlag_SHP

LL_RTC_IsActiveFlag_WUTW

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_WUTW(RTC_TypeDef * RTCx)</code>
Function description	Get Wakeup timer write flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR WUTWF LL_RTC_IsActiveFlag_WUTW

LL_RTC_IsActiveFlag_ALRBW

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALRBW(RTC_TypeDef * RTCx)</code>
Function description	Get Alarm B write flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR ALRBWF LL_RTC_IsActiveFlag_ALRBW

LL_RTC_IsActiveFlag_ALRAW

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALRAW(RTC_TypeDef * RTCx)</code>
Function description	Get Alarm A write flag.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • ISR ALRAWF LL_RTC_IsActiveFlag_ALRAW

LL_RTC_EnableIT_TS

Function name	<code>__STATIC_INLINE void LL_RTC_EnableIT_TS(RTC_TypeDef * RTCx)</code>
Function description	Enable Time-stamp interrupt.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TSIE LL_RTC_EnableIT_TS

LL_RTC_DisableIT_TS

Function name	<code>_STATIC_INLINE void LL_RTC_DisableIT_TS (RTC_TypeDef * RTCx)</code>
Function description	Disable Time-stamp interrupt.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TSIE LL_RTC_DisableIT_TS

LL_RTC_EnableIT_WUT

Function name	<code>_STATIC_INLINE void LL_RTC_EnableIT_WUT (RTC_TypeDef * RTCx)</code>
Function description	Enable Wakeup timer interrupt.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR WUTIE LL_RTC_EnableIT_WUT

LL_RTC_DisableIT_WUT

Function name	<code>_STATIC_INLINE void LL_RTC_DisableIT_WUT (RTC_TypeDef * RTCx)</code>
Function description	Disable Wakeup timer interrupt.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR WUTIE LL_RTC_DisableIT_WUT

LL_RTC_EnableIT_ALRB

Function name	<code>_STATIC_INLINE void LL_RTC_EnableIT_ALRB (RTC_TypeDef * RTCx)</code>
Function description	Enable Alarm B interrupt.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance

Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR ALRBIE LL_RTC_EnableIT_ALRB

LL_RTC_DisableIT_ALRB

Function name	_STATIC_INLINE void LL_RTC_DisableIT_ALRB (RTC_TypeDef * RTCx)
Function description	Disable Alarm B interrupt.
Parameters	<ul style="list-style-type: none">• RTCx: RTC Instance
Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR ALRBIE LL_RTC_DisableIT_ALRB

LL_RTC_EnableIT_ALRA

Function name	_STATIC_INLINE void LL_RTC_EnableIT_ALRA (RTC_TypeDef * RTCx)
Function description	Enable Alarm A interrupt.
Parameters	<ul style="list-style-type: none">• RTCx: RTC Instance
Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR ALRAIE LL_RTC_EnableIT_ALRA

LL_RTC_DisableIT_ALRA

Function name	_STATIC_INLINE void LL_RTC_DisableIT_ALRA (RTC_TypeDef * RTCx)
Function description	Disable Alarm A interrupt.
Parameters	<ul style="list-style-type: none">• RTCx: RTC Instance
Return values	<ul style="list-style-type: none">• None:
Notes	<ul style="list-style-type: none">• Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none">• CR ALRAIE LL_RTC_DisableIT_ALRA

LL_RTC_EnableIT_TAMP

Function name	<code>__STATIC_INLINE void LL_RTC_EnableIT_TAMP (RTC_TypeDef * RTCx)</code>
Function description	Enable all Tamper Interrupt.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMPIE LL_RTC_EnableIT_TAMP

LL_RTC_DisableIT_TAMP

Function name	<code>__STATIC_INLINE void LL_RTC_DisableIT_TAMP (RTC_TypeDef * RTCx)</code>
Function description	Disable all Tamper Interrupt.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMPIE LL_RTC_DisableIT_TAMP

LL_RTC_IsEnabledIT_TS

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_TS (RTC_TypeDef * RTCx)</code>
Function description	Check if Time-stamp interrupt is enabled or not.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR TSIE LL_RTC_IsEnabledIT_TS

LL_RTC_IsEnabledIT_WUT

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_WUT (RTC_TypeDef * RTCx)</code>
Function description	Check if Wakeup timer interrupt is enabled or not.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR WUTIE LL_RTC_IsEnabledIT_WUT

LL_RTC_IsEnabledIT_ALRB

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_ALRB (RTC_TypeDef * RTCx)</code>
Function description	Check if Alarm B interrupt is enabled or not.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR ALRBIE LL_RTC_IsEnabledIT_ALRB

LL_RTC_IsEnabledIT_ALRA

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_ALRA (RTC_TypeDef * RTCx)</code>
Function description	Check if Alarm A interrupt is enabled or not.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR ALRAIE LL_RTC_IsEnabledIT_ALRA

LL_RTC_IsEnabledIT_TAMP

Function name	<code>__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_TAMP (RTC_TypeDef * RTCx)</code>
Function description	Check if all the TAMPER interrupts are enabled or not.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TAFCR TAMPIE LL_RTC_IsEnabledIT_TAMP

LL_RTC_DelInit

Function name	ErrorStatus LL_RTC_DelInit (RTC_TypeDef * RTCx)
Function description	De-Initializes the RTC registers to their default reset values.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: RTC registers are de-initialized – ERROR: RTC registers are not de-initialized
Notes	<ul style="list-style-type: none"> • This function doesn't reset the RTC Clock source and RTC Backup Data registers.

LL_RTC_Init

Function name	ErrorStatus LL_RTC_Init (RTC_TypeDef * RTCx, LL_RTC_InitTypeDef * RTC_InitStruct)
Function description	Initializes the RTC registers according to the specified parameters in RTC_InitStruct.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • RTC_InitStruct: pointer to a LL_RTC_InitTypeDef structure that contains the configuration information for the RTC peripheral.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: RTC registers are initialized – ERROR: RTC registers are not initialized
Notes	<ul style="list-style-type: none"> • The RTC Prescaler register is write protected and can be written in initialization mode only.

LL_RTC_StructInit

Function name	void LL_RTC_StructInit (LL_RTC_InitTypeDef * RTC_InitStruct)
Function description	Set each LL_RTC_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> • RTC_InitStruct: pointer to a LL_RTC_InitTypeDef structure which will be initialized.
Return values	<ul style="list-style-type: none"> • None:

LL_RTC_TIME_Init

Function name	ErrorStatus LL_RTC_TIME_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_TimeTypeDef * RTC_TimeStruct)
Function description	Set the RTC current time.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • RTC_Format: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_RTC_FORMAT_BIN – LL_RTC_FORMAT_BCD • RTC_TimeStruct: pointer to a RTC_TimeTypeDef structure that contains the time configuration information for the RTC.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: RTC Time register is configured – ERROR: RTC Time register is not configured

LL_RTC_TIME_StructInit

Function name	void LL_RTC_TIME_StructInit (LL_RTC_TimeTypeDef * RTC_TimeStruct)
Function description	Set each LL_RTC_TimeTypeDef field to default value (Time = 00h:00min:00sec).

Parameters	<ul style="list-style-type: none"> RTC_TimeStruct: pointer to a LL_RTC_TimeTypeDef structure which will be initialized.
Return values	<ul style="list-style-type: none"> None:

LL_RTC_DATE_Init

Function name	ErrorStatus LL_RTC_DATE_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_DateTypeDef * RTC_DateStruct)
Function description	Set the RTC current date.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance RTC_Format: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_FORMAT_BIN - LL_RTC_FORMAT_BCD RTC_DateStruct: pointer to a RTC_DateTypeDef structure that contains the date configuration information for the RTC.
Return values	<ul style="list-style-type: none"> An: ErrorStatus enumeration value: <ul style="list-style-type: none"> - SUCCESS: RTC Day register is configured - ERROR: RTC Day register is not configured

LL_RTC_DATE_StructInit

Function name	void LL_RTC_DATE_StructInit (LL_RTC_DateTypeDef * RTC_DateStruct)
Function description	Set each LL_RTC_DateTypeDef field to default value (date = Monday, January 01 xx00)
Parameters	<ul style="list-style-type: none"> RTC_DateStruct: pointer to a LL_RTC_DateTypeDef structure which will be initialized.
Return values	<ul style="list-style-type: none"> None:

LL_RTC_ALMA_Init

Function name	ErrorStatus LL_RTC_ALMA_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_AlarmTypeDef * RTC_AlarmStruct)
Function description	Set the RTC Alarm A.
Parameters	<ul style="list-style-type: none"> RTCx: RTC Instance RTC_Format: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_FORMAT_BIN - LL_RTC_FORMAT_BCD RTC_AlarmStruct: pointer to a LL_RTC_AlarmTypeDef structure that contains the alarm configuration parameters.
Return values	<ul style="list-style-type: none"> An: ErrorStatus enumeration value: <ul style="list-style-type: none"> - SUCCESS: ALARMA registers are configured - ERROR: ALARMA registers are not configured
Notes	<ul style="list-style-type: none"> The Alarm register can only be written when the

corresponding Alarm is disabled (Use LL_RTC_ALMA_Disable function).

LL_RTC_ALMB_Init

Function name	ErrorStatus LL_RTC_ALMB_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_AlarmTypeDef * RTC_AlarmStruct)
Function description	Set the RTC Alarm B.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance • RTC_Format: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_RTC_FORMAT_BIN - LL_RTC_FORMAT_BCD • RTC_AlarmStruct: pointer to a LL_RTC_AlarmTypeDef structure that contains the alarm configuration parameters.
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> - SUCCESS: ALARMB registers are configured - ERROR: ALARMB registers are not configured
Notes	<ul style="list-style-type: none"> • The Alarm register can only be written when the corresponding Alarm is disabled (LL_RTC_ALMB_Disable function).

LL_RTC_ALMA_StructInit

Function name	void LL_RTC_ALMA_StructInit (LL_RTC_AlarmTypeDef * RTC_AlarmStruct)
Function description	Set each LL_RTC_AlarmTypeDef of ALARMA field to default value (Time = 00h:00mn:00sec / Day = 1st day of the month/Mask = all fields are masked).
Parameters	<ul style="list-style-type: none"> • RTC_AlarmStruct: pointer to a LL_RTC_AlarmTypeDef structure which will be initialized.
Return values	<ul style="list-style-type: none"> • None:

LL_RTC_ALMB_StructInit

Function name	void LL_RTC_ALMB_StructInit (LL_RTC_AlarmTypeDef * RTC_AlarmStruct)
Function description	Set each LL_RTC_AlarmTypeDef of ALARMA field to default value (Time = 00h:00mn:00sec / Day = 1st day of the month/Mask = all fields are masked).
Parameters	<ul style="list-style-type: none"> • RTC_AlarmStruct: pointer to a LL_RTC_AlarmTypeDef structure which will be initialized.
Return values	<ul style="list-style-type: none"> • None:

LL_RTC_EnterInitMode

Function name	ErrorStatus LL_RTC_EnterInitMode (RTC_TypeDef * RTCx)
---------------	--

Function description	Enters the RTC Initialization mode.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: RTC is in Init mode – ERROR: RTC is not in Init mode
Notes	<ul style="list-style-type: none"> • The RTC Initialization mode is write protected, use the <code>LL_RTC_DisableWriteProtection</code> before calling this function.

LL_RTC_ExitInitMode

Function name	ErrorStatus LL_RTC_ExitInitMode (RTC_TypeDef * RTCx)
Function description	Exit the RTC Initialization mode.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: RTC exited from in Init mode – ERROR: Not applicable
Notes	<ul style="list-style-type: none"> • When the initialization sequence is complete, the calendar restarts counting after 4 RTCCLK cycles. • The RTC Initialization mode is write protected, use the <code>LL_RTC_DisableWriteProtection</code> before calling this function.

LL_RTC_WaitForSynchro

Function name	ErrorStatus LL_RTC_WaitForSynchro (RTC_TypeDef * RTCx)
Function description	Waits until the RTC Time and Day registers (RTC_TR and RTC_DR) are synchronized with RTC APB clock.
Parameters	<ul style="list-style-type: none"> • RTCx: RTC Instance
Return values	<ul style="list-style-type: none"> • An: ErrorStatus enumeration value: <ul style="list-style-type: none"> – SUCCESS: RTC registers are synchronised – ERROR: RTC registers are not synchronised
Notes	<ul style="list-style-type: none"> • The RTC Resynchronization mode is write protected, use the <code>LL_RTC_DisableWriteProtection</code> before calling this function. • To read the calendar through the shadow registers after Calendar initialization, calendar update or after wakeup from low power modes the software must first clear the RSF flag. The software must then wait until it is set again before reading the calendar, which means that the calendar registers have been correctly copied into the RTC_TR and RTC_DR shadow registers.

63.3 RTC Firmware driver defines

63.3.1 RTC

ALARM OUTPUT

`LL_RTC_ALARMOUT_DISABLE` Output disabled

LL_RTC_ALARMOUT_ALMA	Alarm A output enabled
LL_RTC_ALARMOUT_ALMB	Alarm B output enabled
LL_RTC_ALARMOUT_WAKEUP	Wakeup output enabled

ALARM OUTPUT TYPE

LL_RTC_ALARM_OUTPUTTYPE_OPENDRAIN	RTC_ALARM, when mapped on PC13, is open-drain output
LL_RTC_ALARM_OUTPUTTYPE_PUSHPULL	RTC_ALARM, when mapped on PC13, is push-pull output

ALARMA MASK

LL_RTC_ALMA_MASK_NONE	No masks applied on Alarm A
LL_RTC_ALMA_MASK_DATEWEEKDAY	Date/day do not care in Alarm A comparison
LL_RTC_ALMA_MASK_HOURS	Hours do not care in Alarm A comparison
LL_RTC_ALMA_MASK_MINUTES	Minutes do not care in Alarm A comparison
LL_RTC_ALMA_MASK_SECONDS	Seconds do not care in Alarm A comparison
LL_RTC_ALMA_MASK_ALL	Masks all

ALARMA TIME FORMAT

LL_RTC_ALMA_TIME_FORMAT_AM	AM or 24-hour format
LL_RTC_ALMA_TIME_FORMAT_PM	PM

RTC Alarm A Date WeekDay

LL_RTC_ALMA_DATEWEEKDAYSEL_DATE	Alarm A Date is selected
LL_RTC_ALMA_DATEWEEKDAYSEL_WEEKDAY	Alarm A WeekDay is selected

ALARMB MASK

LL_RTC_ALMB_MASK_NONE	No masks applied on Alarm B
LL_RTC_ALMB_MASK_DATEWEEKDAY	Date/day do not care in Alarm B comparison
LL_RTC_ALMB_MASK_HOURS	Hours do not care in Alarm B comparison
LL_RTC_ALMB_MASK_MINUTES	Minutes do not care in Alarm B comparison
LL_RTC_ALMB_MASK_SECONDS	Seconds do not care in Alarm B comparison
LL_RTC_ALMB_MASK_ALL	Masks all

ALARMB TIME FORMAT

LL_RTC_ALMB_TIME_FORMAT_AM	AM or 24-hour format
LL_RTC_ALMB_TIME_FORMAT_PM	PM

RTC Alarm B Date WeekDay

LL_RTC_ALMB_DATEWEEKDAYSEL_DATE	Alarm B Date is selected
LL_RTC_ALMB_DATEWEEKDAYSEL_WEEKDAY	Alarm B WeekDay is selected

BACKUP

LL_RTC_BKP_DR0	
LL_RTC_BKP_DR1	

LL_RTC_BKP_DR2
LL_RTC_BKP_DR3
LL_RTC_BKP_DR4
LL_RTC_BKP_DR5
LL_RTC_BKP_DR6
LL_RTC_BKP_DR7
LL_RTC_BKP_DR8
LL_RTC_BKP_DR9
LL_RTC_BKP_DR10
LL_RTC_BKP_DR11
LL_RTC_BKP_DR12
LL_RTC_BKP_DR13
LL_RTC_BKP_DR14
LL_RTC_BKP_DR15
LL_RTC_BKP_DR16
LL_RTC_BKP_DR17
LL_RTC_BKP_DR18
LL_RTC_BKP_DR19
LL_RTC_BKP_DR20
LL_RTC_BKP_DR21
LL_RTC_BKP_DR22
LL_RTC_BKP_DR23
LL_RTC_BKP_DR24
LL_RTC_BKP_DR25
LL_RTC_BKP_DR26
LL_RTC_BKP_DR27
LL_RTC_BKP_DR28
LL_RTC_BKP_DR29
LL_RTC_BKP_DR30
LL_RTC_BKP_DR31

Calibration pulse insertion

LL_RTC_CALIB_INSERTPULSE_NONE	No RTCCLK pulses are added
LL_RTC_CALIB_INSERTPULSE_SET	One RTCCLK pulse is effectively inserted every 2×10^{11} pulses (frequency increased by 488.5 ppm)

Calibration output

LL_RTC_CALIB_OUTPUT_NONE	Calibration output disabled
--------------------------	-----------------------------

LL_RTC_CALIB_OUTPUT_1HZ	Calibration output is 1 Hz
LL_RTC_CALIB_OUTPUT_512HZ	Calibration output is 512 Hz

Calibration period

LL_RTC_CALIB_PERIOD_32SEC	Use a 32-second calibration cycle period
LL_RTC_CALIB_PERIOD_16SEC	Use a 16-second calibration cycle period
LL_RTC_CALIB_PERIOD_8SEC	Use a 8-second calibration cycle period

Coarse digital calibration sign

LL_RTC_CALIB_SIGN_POSITIVE	Positive calibration: calendar update frequency is increased
LL_RTC_CALIB_SIGN_NEGATIVE	Negative calibration: calendar update frequency is decreased

FORMAT

LL_RTC_FORMAT_BIN	Binary data format
LL_RTC_FORMAT_BCD	BCD data format

Get Flags Defines

LL_RTC_ISR_RECALPF
LL_RTC_ISR_TAMP3F
LL_RTC_ISR_TAMP2F
LL_RTC_ISR_TAMP1F
LL_RTC_ISR_TSOVF
LL_RTC_ISR_TSOF
LL_RTC_ISR_WUTF
LL_RTC_ISR_ALRBF
LL_RTC_ISR_ALRAF
LL_RTC_ISR_INITF
LL_RTC_ISR_RSF
LL_RTC_ISR_INITS
LL_RTC_ISR_SHPF
LL_RTC_ISR_WUTWF
LL_RTC_ISR_ALRBWF
LL_RTC_ISR_ALRAWF

HOUR FORMAT

LL_RTC_HOURFORMAT_24HOUR	24 hour/day format
LL_RTC_HOURFORMAT_AMPM	AM/PM hour format

IT Defines

LL_RTC_CR_TSIE
LL_RTC_CR_WUTIE

`LL_RTC_CR_ALRBIE`

`LL_RTC_CR_ALRAIE`

`LL_RTC_TAFCR_TAMPIE`

MONTH

<code>LL_RTC_MONTH_JANUARY</code>	January
<code>LL_RTC_MONTH_FEBRUARY</code>	February
<code>LL_RTC_MONTH_MARCH</code>	March
<code>LL_RTC_MONTH_APRIIL</code>	April
<code>LL_RTC_MONTH_MAY</code>	May
<code>LL_RTC_MONTH_JUNE</code>	June
<code>LL_RTC_MONTH_JULY</code>	July
<code>LL_RTC_MONTH_AUGUST</code>	August
<code>LL_RTC_MONTH_SEPTEMBER</code>	September
<code>LL_RTC_MONTH_OCTOBER</code>	October
<code>LL_RTC_MONTH_NOVEMBER</code>	November
<code>LL_RTC_MONTH_DECEMBER</code>	December

OUTPUT POLARITY PIN

<code>LL_RTC_OUTPUTPOLARITY_PIN_HIGH</code>	Pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL)
<code>LL_RTC_OUTPUTPOLARITY_PIN_LOW</code>	Pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL)

SHIFT SECOND

`LL_RTC_SHIFT_SECOND_DELAY`

`LL_RTC_SHIFT_SECOND_ADVANCE`

TAMPER

`LL_RTC_TAMPER_1` RTC_TAMP1 input detection

`LL_RTC_TAMPER_2` RTC_TAMP2 input detection

`LL_RTC_TAMPER_3` RTC_TAMP3 input detection

TAMPER ACTIVE LEVEL

`LL_RTC_TAMPER_ACTIVELEVEL_TAMP1` RTC_TAMP1 input falling edge (if TAMPFLT = 00) or staying high (if TAMPFLT != 00) triggers a tamper detection event

`LL_RTC_TAMPER_ACTIVELEVEL_TAMP2` RTC_TAMP2 input falling edge (if TAMPFLT = 00) or staying high (if TAMPFLT != 00) triggers a tamper detection event

`LL_RTC_TAMPER_ACTIVELEVEL_TAMP3` RTC_TAMP3 input falling edge (if TAMPFLT = 00) or staying high (if TAMPFLT != 00) triggers a tamper detection

event

TAMPER DURATION

LL_RTC_TAMPER_DURATION_1RTCCLK	Tamper pins are pre-charged before sampling during 1 RTCCLK cycle
LL_RTC_TAMPER_DURATION_2RTCCLK	Tamper pins are pre-charged before sampling during 2 RTCCLK cycles
LL_RTC_TAMPER_DURATION_4RTCCLK	Tamper pins are pre-charged before sampling during 4 RTCCLK cycles
LL_RTC_TAMPER_DURATION_8RTCCLK	Tamper pins are pre-charged before sampling during 8 RTCCLK cycles

TAMPER FILTER

LL_RTC_TAMPER_FILTER_DISABLE	Tamper filter is disabled
LL_RTC_TAMPER_FILTER_2SAMPLE	Tamper is activated after 2 consecutive samples at the active level
LL_RTC_TAMPER_FILTER_4SAMPLE	Tamper is activated after 4 consecutive samples at the active level
LL_RTC_TAMPER_FILTER_8SAMPLE	Tamper is activated after 8 consecutive samples at the active level.

TAMPER MASK

LL_RTC_TAMPER_MASK_TAMPER1	Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers are not erased
LL_RTC_TAMPER_MASK_TAMPER2	Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers are not erased.
LL_RTC_TAMPER_MASK_TAMPER3	Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers are not erased

TAMPER NO ERASE

LL_RTC_TAMPER_NOERASE_TAMPER1	Tamper 1 event does not erase the backup registers.
LL_RTC_TAMPER_NOERASE_TAMPER2	Tamper 2 event does not erase the backup registers.
LL_RTC_TAMPER_NOERASE_TAMPER3	Tamper 3 event does not erase the backup registers.

TAMPER SAMPLING FREQUENCY DIVIDER

LL_RTC_TAMPER_SAMPLFREQDIV_32768	Each of the tamper inputs are sampled with a frequency = RTCCLK / 32768
LL_RTC_TAMPER_SAMPLFREQDIV_16384	Each of the tamper inputs are sampled with a frequency = RTCCLK / 16384
LL_RTC_TAMPER_SAMPLFREQDIV_8192	Each of the tamper inputs are sampled with a frequency = RTCCLK / 8192
LL_RTC_TAMPER_SAMPLFREQDIV_4096	Each of the tamper inputs are sampled

LL_RTC_TAMPER_SAMPLFREQDIV_2048	with a frequency = RTCCLK / 4096
LL_RTC_TAMPER_SAMPLFREQDIV_1024	Each of the tamper inputs are sampled with a frequency = RTCCLK / 2048
LL_RTC_TAMPER_SAMPLFREQDIV_512	Each of the tamper inputs are sampled with a frequency = RTCCLK / 1024
LL_RTC_TAMPER_SAMPLFREQDIV_256	Each of the tamper inputs are sampled with a frequency = RTCCLK / 512

TIMESTAMP EDGE

LL_RTC_TIMESTAMP_EDGE_RISING	RTC_TS input rising edge generates a timestamp event
LL_RTC_TIMESTAMP_EDGE_FALLING	RTC_TS input falling edge generates a timestamp even

TIME FORMAT

LL_RTC_TIME_FORMAT_AM_OR_24	AM or 24-hour format
LL_RTC_TIME_FORMAT_PM	PM

TIMESTAMP TIME FORMAT

LL_RTC_TS_TIME_FORMAT_AM	AM or 24-hour format
LL_RTC_TS_TIME_FORMAT_PM	PM

WAKEUP CLOCK DIV

LL_RTC_WAKEUPCLOCK_DIV_16	RTC/16 clock is selected
LL_RTC_WAKEUPCLOCK_DIV_8	RTC/8 clock is selected
LL_RTC_WAKEUPCLOCK_DIV_4	RTC/4 clock is selected
LL_RTC_WAKEUPCLOCK_DIV_2	RTC/2 clock is selected
LL_RTC_WAKEUPCLOCK_CKSPRE	ck_spre (usually 1 Hz) clock is selected
LL_RTC_WAKEUPCLOCK_CKSPRE_WUT	ck_spre (usually 1 Hz) clock is selected and 2exp16 is added to the WUT counter value

WEEK DAY

LL_RTC_WEEKDAY_MONDAY	Monday
LL_RTC_WEEKDAY_TUESDAY	Tuesday
LL_RTC_WEEKDAY_WEDNESDAY	Wednesday
LL_RTC_WEEKDAY_THURSDAY	Thrusday
LL_RTC_WEEKDAY_FRIDAY	Friday
LL_RTC_WEEKDAY_SATURDAY	Saturday
LL_RTC_WEEKDAY_SUNDAY	Sunday

Convert helper Macros

`_LL_RTC_CONVERT_BIN2BCD` **Description:**

- Helper macro to convert a value from 2 digit

decimal format to BCD format.

Parameters:

- `__VALUE__`: Byte to be converted

Return value:

- Converted: byte

`__LL_RTC_CONVERT_BCD2BIN`

Description:

- Helper macro to convert a value from BCD format to 2 digit decimal format.

Parameters:

- `__VALUE__`: BCD value to be converted

Return value:

- Converted: byte

Date helper Macros

`__LL_RTC_GET_WEEKDAY`

Description:

- Helper macro to retrieve weekday.

Parameters:

- `__RTC_DATE__`: Date returned by

Return value:

- Returned: value can be one of the following values:
 - `LL_RTC_WEEKDAY_MONDAY`
 - `LL_RTC_WEEKDAY_TUESDAY`
 - `LL_RTC_WEEKDAY_WEDNESDAY`
 - `LL_RTC_WEEKDAY_THURSDAY`
 - `LL_RTC_WEEKDAY_FRIDAY`
 - `LL_RTC_WEEKDAY_SATURDAY`
 - `LL_RTC_WEEKDAY_SUNDAY`

`__LL_RTC_GET_YEAR`

Description:

- Helper macro to retrieve Year in BCD format.

Parameters:

- `__RTC_DATE__`: Value returned by

Return value:

- Year: in BCD format (0x00 . . . 0x99)

`__LL_RTC_GET_MONTH`

Description:

- Helper macro to retrieve Month in BCD format.

Parameters:

- `__RTC_DATE__`: Value returned by

Return value:

- Returned: value can be one of the following values:
 - `LL_RTC_MONTH_JANUARY`

- LL_RTC_MONTH_FEBRUARY
- LL_RTC_MONTH_MARCH
- LL_RTC_MONTH_APRI
- LL_RTC_MONTH_MAY
- LL_RTC_MONTH_JUNE
- LL_RTC_MONTH_JULY
- LL_RTC_MONTH_AUGUST
- LL_RTC_MONTH_SEPTEMBER
- LL_RTC_MONTH_OCTOBER
- LL_RTC_MONTH_NOVEMBER
- LL_RTC_MONTH_DECEMBER

__LL_RTC_GET_DAY**Description:**

- Helper macro to retrieve Day in BCD format.

Parameters:

- __RTC_DATE__: Value returned by

Return value:

- Day: in BCD format (0x01 . . . 0x31)

Time helper Macros**__LL_RTC_GET_HOUR****Description:**

- Helper macro to retrieve hour in BCD format.

Parameters:

- __RTC_TIME__: RTC time returned by

Return value:

- Hours: in BCD format (0x01 . . . 0x12 or between Min_Data=0x00 and Max_Data=0x23)

__LL_RTC_GET_MINUTE**Description:**

- Helper macro to retrieve minute in BCD format.

Parameters:

- __RTC_TIME__: RTC time returned by

Return value:

- Minutes: in BCD format (0x00 . . . 0x59)

__LL_RTC_GET_SECOND**Description:**

- Helper macro to retrieve second in BCD format.

Parameters:

- __RTC_TIME__: RTC time returned by

Return value:

- Seconds: in format (0x00 . . . 0x59)

Common Write and read registers Macros**LL_RTC_WriteReg****Description:**

- Write a value in RTC register.

Parameters:

- __INSTANCE__: RTC Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_RTC_ReadReg

Description:

- Read a value in RTC register.

Parameters:

- __INSTANCE__: RTC Instance
- __REG__: Register to be read

Return value:

- Register: value

64 LL SPI Generic Driver

64.1 SPI Firmware driver registers structures

64.1.1 LL_SPI_InitTypeDef

Data Fields

- *uint32_t TransferDirection*
- *uint32_t Mode*
- *uint32_t DataWidth*
- *uint32_t ClockPolarity*
- *uint32_t ClockPhase*
- *uint32_t NSS*
- *uint32_t BaudRate*
- *uint32_t BitOrder*
- *uint32_t CRCCalculation*
- *uint32_t CRCPoly*

Field Documentation

- ***uint32_t LL_SPI_InitTypeDef::TransferDirection***
Specifies the SPI unidirectional or bidirectional data mode. This parameter can be a value of [**SPI_LL_EC_TRANSFER_MODE**](#). This feature can be modified afterwards using unitary function [**LL_SPI_SetTransferDirection\(\)**](#).
- ***uint32_t LL_SPI_InitTypeDef::Mode***
Specifies the SPI mode (Master/Slave). This parameter can be a value of [**SPI_LL_EC_MODE**](#). This feature can be modified afterwards using unitary function [**LL_SPI_SetMode\(\)**](#).
- ***uint32_t LL_SPI_InitTypeDef::DataWidth***
Specifies the SPI data width. This parameter can be a value of [**SPI_LL_EC_DATAWIDTH**](#). This feature can be modified afterwards using unitary function [**LL_SPI_SetDataWidth\(\)**](#).
- ***uint32_t LL_SPI_InitTypeDef::ClockPolarity***
Specifies the serial clock steady state. This parameter can be a value of [**SPI_LL_EC_POLARITY**](#). This feature can be modified afterwards using unitary function [**LL_SPI_SetClockPolarity\(\)**](#).
- ***uint32_t LL_SPI_InitTypeDef::ClockPhase***
Specifies the clock active edge for the bit capture. This parameter can be a value of [**SPI_LL_EC_PHASE**](#). This feature can be modified afterwards using unitary function [**LL_SPI_SetClockPhase\(\)**](#).
- ***uint32_t LL_SPI_InitTypeDef::NSS***
Specifies whether the NSS signal is managed by hardware (NSS pin) or by software using the SSI bit. This parameter can be a value of [**SPI_LL_EC_NSS_MODE**](#). This feature can be modified afterwards using unitary function [**LL_SPI_SetNSSMode\(\)**](#).
- ***uint32_t LL_SPI_InitTypeDef::BaudRate***
Specifies the BaudRate prescaler value which will be used to configure the transmit and receive SCK clock. This parameter can be a value of [**SPI_LL_EC_BAUDRATEPRESCALER**](#).
Note: The communication clock is derived from the master clock. The slave clock does not need to be set. This feature can be modified afterwards using unitary function [**LL_SPI_SetBaudRatePrescaler\(\)**](#).

- **`uint32_t LL_SPI_InitTypeDef::BitOrder`**
Specifies whether data transfers start from MSB or LSB bit. This parameter can be a value of `SPI_LL_EC_BIT_ORDER`. This feature can be modified afterwards using unitary function `LL_SPI_SetTransferBitOrder()`.
- **`uint32_t LL_SPI_InitTypeDef::CRCCalculation`**
Specifies if the CRC calculation is enabled or not. This parameter can be a value of `SPI_LL_EC_CRC_CALCULATION`. This feature can be modified afterwards using unitary functions `LL_SPI_EnableCRC()` and `LL_SPI_DisableCRC()`.
- **`uint32_t LL_SPI_InitTypeDef::CRCPoly`**
Specifies the polynomial used for the CRC calculation. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFFFF. This feature can be modified afterwards using unitary function `LL_SPI_SetCRCPolynomial()`.

64.2 SPI Firmware driver API description

64.2.1 Detailed description of functions

LL_SPI_Enable

Function name	<code>__STATIC_INLINE void LL_SPI_Enable (SPI_TypeDef * SPIx)</code>
Function description	Enable SPI peripheral.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 SPE LL_SPI_Enable

LL_SPI_Disable

Function name	<code>__STATIC_INLINE void LL_SPI_Disable (SPI_TypeDef * SPIx)</code>
Function description	Disable SPI peripheral.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • When disabling the SPI, follow the procedure described in the Reference Manual.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 SPE LL_SPI_Disable

LL_SPI_IsEnabled

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsEnabled (SPI_TypeDef * SPIx)</code>
Function description	Check if SPI peripheral is enabled.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross	<ul style="list-style-type: none"> • CR1 SPE LL_SPI_IsEnabled

reference:

LL_SPI_SetMode

Function name	<code>__STATIC_INLINE void LL_SPI_SetMode (SPI_TypeDef * SPIx, uint32_t Mode)</code>
Function description	Set SPI operation mode to Master or Slave.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • Mode: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_MODE_MASTER - LL_SPI_MODE_SLAVE
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit should not be changed when communication is ongoing.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 MSTR LL_SPI_SetMode • CR1 SSI LL_SPI_SetMode

LL_SPI_GetMode

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_GetMode (SPI_TypeDef * SPIx)</code>
Function description	Get SPI operation mode (Master or Slave)
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_MODE_MASTER - LL_SPI_MODE_SLAVE
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 MSTR LL_SPI_GetMode • CR1 SSI LL_SPI_GetMode

LL_SPI_SetStandard

Function name	<code>__STATIC_INLINE void LL_SPI_SetStandard (SPI_TypeDef * SPIx, uint32_t Standard)</code>
Function description	Set serial protocol used.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • Standard: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_PROTOCOL_MOTOROLA - LL_SPI_PROTOCOL_TI
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit should be written only when SPI is disabled (SPE = 0) for correct operation.
Reference Manual to LL API cross	<ul style="list-style-type: none"> • CR2 FRF LL_SPI_SetStandard

reference:

LL_SPI_GetStandard

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_GetStandard (SPI_TypeDef * SPIx)</code>
Function description	Get serial protocol used.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_PROTOCOL_MOTOROLA - LL_SPI_PROTOCOL_TI
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 FRF LL_SPI_GetStandard

LL_SPI_SetClockPhase

Function name	<code>__STATIC_INLINE void LL_SPI_SetClockPhase (SPI_TypeDef * SPIx, uint32_t ClockPhase)</code>
Function description	Set clock phase.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • ClockPhase: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_PHASE_1EDGE - LL_SPI_PHASE_2EDGE
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit should not be changed when communication is ongoing. This bit is not used in SPI TI mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 CPHA LL_SPI_SetClockPhase

LL_SPI_GetClockPhase

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_GetClockPhase (SPI_TypeDef * SPIx)</code>
Function description	Get clock phase.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_PHASE_1EDGE - LL_SPI_PHASE_2EDGE
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 CPHA LL_SPI_GetClockPhase

LL_SPI_SetClockPolarity

Function name	<code>_STATIC_INLINE void LL_SPI_SetClockPolarity(SPI_TypeDef * SPIx, uint32_t ClockPolarity)</code>
Function description	Set clock polarity.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • ClockPolarity: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_SPI_POLARITY_LOW – LL_SPI_POLARITY_HIGH
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit should not be changed when communication is ongoing. This bit is not used in SPI TI mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 CPOL LL_SPI_SetClockPolarity

LL_SPI_GetClockPolarity

Function name	<code>_STATIC_INLINE uint32_t LL_SPI_GetClockPolarity(SPI_TypeDef * SPIx)</code>
Function description	Get clock polarity.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_SPI_POLARITY_LOW – LL_SPI_POLARITY_HIGH
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 CPOL LL_SPI_GetClockPolarity

LL_SPI_SetBaudRatePrescaler

Function name	<code>_STATIC_INLINE void LL_SPI_SetBaudRatePrescaler(SPI_TypeDef * SPIx, uint32_t BaudRate)</code>
Function description	Set baud rate prescaler.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • BaudRate: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_SPI_BAUDRATEPRESCALER_DIV2 – LL_SPI_BAUDRATEPRESCALER_DIV4 – LL_SPI_BAUDRATEPRESCALER_DIV8 – LL_SPI_BAUDRATEPRESCALER_DIV16 – LL_SPI_BAUDRATEPRESCALER_DIV32 – LL_SPI_BAUDRATEPRESCALER_DIV64 – LL_SPI_BAUDRATEPRESCALER_DIV128 – LL_SPI_BAUDRATEPRESCALER_DIV256
Return values	<ul style="list-style-type: none"> • None:

Notes	<ul style="list-style-type: none"> These bits should not be changed when communication is ongoing. SPI BaudRate = fPCLK/Prescaler.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 BR LL_SPI_SetBaudRatePrescaler

LL_SPI_GetBaudRatePrescaler

Function name	<code>_STATIC_INLINE uint32_t LL_SPI_GetBaudRatePrescaler(SPI_TypeDef * SPIx)</code>
Function description	Get baud rate prescaler.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> LL_SPI_BAUDRATEPRESCALER_DIV2 LL_SPI_BAUDRATEPRESCALER_DIV4 LL_SPI_BAUDRATEPRESCALER_DIV8 LL_SPI_BAUDRATEPRESCALER_DIV16 LL_SPI_BAUDRATEPRESCALER_DIV32 LL_SPI_BAUDRATEPRESCALER_DIV64 LL_SPI_BAUDRATEPRESCALER_DIV128 LL_SPI_BAUDRATEPRESCALER_DIV256
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 BR LL_SPI_GetBaudRatePrescaler

LL_SPI_SetTransferBitOrder

Function name	<code>_STATIC_INLINE void LL_SPI_SetTransferBitOrder(SPI_TypeDef * SPIx, uint32_t BitOrder)</code>
Function description	Set transfer bit order.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance BitOrder: This parameter can be one of the following values: <ul style="list-style-type: none"> LL_SPI_LSB_FIRST LL_SPI_MSB_FIRST
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> This bit should not be changed when communication is ongoing. This bit is not used in SPI TI mode.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 LSBFIRST LL_SPI_SetTransferBitOrder

LL_SPI_GetTransferBitOrder

Function name	<code>_STATIC_INLINE uint32_t LL_SPI_GetTransferBitOrder(SPI_TypeDef * SPIx)</code>
Function description	Get transfer bit order.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance

Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_LSB_FIRST - LL_SPI_MSB_FIRST
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 LSBFIRST LL_SPI_SetTransferDirection

LL_SPI_SetTransferDirection

Function name	<code>__STATIC_INLINE void LL_SPI_SetTransferDirection(SPI_TypeDef * SPIx, uint32_t TransferDirection)</code>
Function description	Set transfer direction mode.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance TransferDirection: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_FULL_DUPLEX - LL_SPI_SIMPLEX_RX - LL_SPI_HALF_DUPLEX_RX - LL_SPI_HALF_DUPLEX_TX
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> For Half-Duplex mode, Rx Direction is set by default. In master mode, the MOSI pin is used and in slave mode, the MISO pin is used for Half-Duplex.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 RXONLY LL_SPI_SetTransferDirection CR1 BIDIMODE LL_SPI_SetTransferDirection CR1 BIDIOE LL_SPI_SetTransferDirection

LL_SPI_GetTransferDirection

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_GetTransferDirection(SPI_TypeDef * SPIx)</code>
Function description	Get transfer direction mode.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_FULL_DUPLEX - LL_SPI_SIMPLEX_RX - LL_SPI_HALF_DUPLEX_RX - LL_SPI_HALF_DUPLEX_TX
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 RXONLY LL_SPI_SetTransferDirection CR1 BIDIMODE LL_SPI_SetTransferDirection CR1 BIDIOE LL_SPI_SetTransferDirection

LL_SPI_SetDataWidth

Function name	<code>__STATIC_INLINE void LL_SPI_SetDataWidth(SPI_TypeDef * SPIx, uint32_t DataWidth)</code>
Function description	Set frame data width.

Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • DataWidth: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_SPI_DATAWIDTH_8BIT – LL_SPI_DATAWIDTH_16BIT
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 DFF LL_SPI_SetDataWidth

LL_SPI_GetDataWidth

Function name	<code>_STATIC_INLINE uint32_t LL_SPI_GetDataWidth(SPI_TypeDef * SPIx)</code>
Function description	Get frame data width.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_SPI_DATAWIDTH_8BIT – LL_SPI_DATAWIDTH_16BIT
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 DFF LL_SPI_GetDataWidth

LL_SPI_EnableCRC

Function name	<code>_STATIC_INLINE void LL_SPI_EnableCRC(SPI_TypeDef * SPIx)</code>
Function description	Enable CRC.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit should be written only when SPI is disabled (SPE = 0) for correct operation.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 CRCEN LL_SPI_EnableCRC

LL_SPI_DisableCRC

Function name	<code>_STATIC_INLINE void LL_SPI_DisableCRC(SPI_TypeDef * SPIx)</code>
Function description	Disable CRC.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit should be written only when SPI is disabled (SPE = 0) for correct operation.

Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR1 CRCEN LL_SPI_DisableCRC
---	---

LL_SPI_IsEnabledCRC

Function name Function description Parameters Return values Notes Reference Manual to LL API cross reference:	<pre style="font-family: monospace; font-size: 10pt; margin: 0;">_STATIC_INLINE uint32_t LL_SPI_IsEnabledCRC (SPI_TypeDef * SPIx)</pre> <p>Check if CRC is enabled.</p> <ul style="list-style-type: none"> • SPIx: SPI Instance • State: of bit (1 or 0). • This bit should be written only when SPI is disabled (SPE = 0) for correct operation. <ul style="list-style-type: none"> • CR1 CRCEN LL_SPI_IsEnabledCRC
--	--

LL_SPI_SetCRCNext

Function name Function description Parameters Return values Notes Reference Manual to LL API cross reference:	<pre style="font-family: monospace; font-size: 10pt; margin: 0;">_STATIC_INLINE void LL_SPI_SetCRCNext (SPI_TypeDef * SPIx)</pre> <p>Set CRCNext to transfer CRC on the line.</p> <ul style="list-style-type: none"> • SPIx: SPI Instance • None: • This bit has to be written as soon as the last data is written in the SPIx_DR register. <ul style="list-style-type: none"> • CR1 CRCNEXT LL_SPI_SetCRCNext
--	---

LL_SPI_SetCRCPolynomial

Function name Function description Parameters Return values Reference Manual to LL API cross reference:	<pre style="font-family: monospace; font-size: 10pt; margin: 0;">_STATIC_INLINE void LL_SPI_SetCRCPolynomial (SPI_TypeDef * SPIx, uint32_t CRCPoly)</pre> <p>Set polynomial for CRC calculation.</p> <ul style="list-style-type: none"> • SPIx: SPI Instance • CRCPoly: This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFFFF • None: <ul style="list-style-type: none"> • CRCPOLY CRCPOLY LL_SPI_SetCRCPolynomial
---	---

LL_SPI_GetCRCPolynomial

Function name	<pre style="font-family: monospace; font-size: 10pt; margin: 0;">_STATIC_INLINE uint32_t LL_SPI_GetCRCPolynomial</pre>
---------------	---

(SPI_TypeDef * SPIx)

Function description	Get polynomial for CRC calculation.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value is a number between Min_Data = 0x00 and Max_Data = 0xFFFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CRCPR CRCPOLY LL_SPI_GetCRCPolynomial

LL_SPI_GetRxCRC

Function name	_STATIC_INLINE uint32_t LL_SPI_GetRxCRC (SPI_TypeDef * SPIx)
Function description	Get Rx CRC.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value is a number between Min_Data = 0x00 and Max_Data = 0xFFFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RXCRCR RXCRC LL_SPI_GetRxCRC

LL_SPI_GetTxCRC

Function name	_STATIC_INLINE uint32_t LL_SPI_GetTxCRC (SPI_TypeDef * SPIx)
Function description	Get Tx CRC.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • Returned: value is a number between Min_Data = 0x00 and Max_Data = 0xFFFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • TXCRCR TXCRC LL_SPI_GetTxCRC

LL_SPI_SetNSSMode

Function name	_STATIC_INLINE void LL_SPI_SetNSSMode (SPI_TypeDef * SPIx, uint32_t NSS)
Function description	Set NSS mode.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance • NSS: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_SPI_NSS_SOFT - LL_SPI_NSS_HARD_INPUT - LL_SPI_NSS_HARD_OUTPUT
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • LL_SPI_NSS_SOFT Mode is not used in SPI TI mode.

- Reference Manual to
LL API cross
reference:
- CR1 SSM LL_SPI_SetNSSMode
 - CR2 SSOE LL_SPI_SetNSSMode

LL_SPI_GetNSSMode

Function name **`_STATIC_INLINE uint32_t LL_SPI_GetNSSMode(SPI_TypeDef * SPIx)`**

Function description Get NSS mode.

Parameters • **SPIx:** SPI Instance

Return values • **Returned:** value can be one of the following values:
 – LL_SPI_NSS_SOFT
 – LL_SPI_NSS_HARD_INPUT
 – LL_SPI_NSS_HARD_OUTPUT

- Reference Manual to
LL API cross
reference:
- CR1 SSM LL_SPI_GetNSSMode
 - CR2 SSOE LL_SPI_GetNSSMode

LL_SPI_IsActiveFlag_RXNE

Function name **`_STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_RXNE(SPI_TypeDef * SPIx)`**

Function description Check if Rx buffer is not empty.

Parameters • **SPIx:** SPI Instance

Return values • **State:** of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- SR RXNE LL_SPI_IsActiveFlag_RXNE

LL_SPI_IsActiveFlag_TXE

Function name **`_STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_TXE(SPI_TypeDef * SPIx)`**

Function description Check if Tx buffer is empty.

Parameters • **SPIx:** SPI Instance

Return values • **State:** of bit (1 or 0).

- Reference Manual to
LL API cross
reference:
- SR TXE LL_SPI_IsActiveFlag_TXE

LL_SPI_IsActiveFlag_CRCERR

Function name **`_STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_CRCERR(SPI_TypeDef * SPIx)`**

Function description Get CRC error flag.

Parameters • **SPIx:** SPI Instance

Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR CRCERR LL_SPI_IsActiveFlag_CRCERR

LL_SPI_IsActiveFlag_MODF

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_MODF(SPI_TypeDef * SPIx)</code>
Function description	Get mode fault error flag.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR MODF LL_SPI_IsActiveFlag_MODF

LL_SPI_IsActiveFlag_OVR

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_OVR(SPI_TypeDef * SPIx)</code>
Function description	Get overrun error flag.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR OVR LL_SPI_IsActiveFlag_OVR

LL_SPI_IsActiveFlag_BSY

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_BSY(SPI_TypeDef * SPIx)</code>
Function description	Get busy flag.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Notes	<ul style="list-style-type: none"> The BSY flag is cleared under any one of the following conditions: <ul style="list-style-type: none"> -When the SPI is correctly disabled -When a fault is detected in Master mode (MODF bit set to 1) -In Master mode, when it finishes a data transmission and no new data is ready to be sent -In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SR BSY LL_SPI_IsActiveFlag_BSY

LL_SPI_IsActiveFlag_FRE

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_FRE (SPI_TypeDef * SPIx)</code>
Function description	Get frame format error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR FRE LL_SPI_IsActiveFlag_FRE

LL_SPI_ClearFlag_CRCERR

Function name	<code>__STATIC_INLINE void LL_SPI_ClearFlag_CRCERR (SPI_TypeDef * SPIx)</code>
Function description	Clear CRC error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR CRCERR LL_SPI_ClearFlag_CRCERR

LL_SPI_ClearFlag_MODF

Function name	<code>__STATIC_INLINE void LL_SPI_ClearFlag_MODF (SPI_TypeDef * SPIx)</code>
Function description	Clear mode fault error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Clearing this flag is done by a read access to the SPIx_SR register followed by a write access to the SPIx_CR1 register
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR MODF LL_SPI_ClearFlag_MODF

LL_SPI_ClearFlag_OVR

Function name	<code>__STATIC_INLINE void LL_SPI_ClearFlag_OVR (SPI_TypeDef * SPIx)</code>
Function description	Clear overrun error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Clearing this flag is done by a read access to the SPIx_DR register followed by a read access to the SPIx_SR register
Reference Manual to	<ul style="list-style-type: none"> • SR OVR LL_SPI_ClearFlag_OVR

LL API cross
reference:

LL_SPI_ClearFlag_FRE

Function name	<code>_STATIC_INLINE void LL_SPI_ClearFlag_FRE (SPI_TypeDef * SPIx)</code>
Function description	Clear frame format error flag.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Clearing this flag is done by reading SPIx_SR register
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • SR FRE LL_SPI_ClearFlag_FRE

LL_SPI_EnableIT_ERR

Function name	<code>_STATIC_INLINE void LL_SPI_EnableIT_ERR (SPI_TypeDef * SPIx)</code>
Function description	Enable error interrupt.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ERRIE LL_SPI_EnableIT_ERR

LL_SPI_EnableIT_RXNE

Function name	<code>_STATIC_INLINE void LL_SPI_EnableIT_RXNE (SPI_TypeDef * SPIx)</code>
Function description	Enable Rx buffer not empty interrupt.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 RXNEIE LL_SPI_EnableIT_RXNE

LL_SPI_EnableIT_TXE

Function name	<code>_STATIC_INLINE void LL_SPI_EnableIT_TXE (SPI_TypeDef * SPIx)</code>
Function description	Enable Tx buffer empty interrupt.

Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	CR2 TXEIE LL_SPI_EnableIT_TXE

LL_SPI_DisableIT_ERR

Function name	<code>_STATIC_INLINE void LL_SPI_DisableIT_ERR (SPI_TypeDef * SPIx)</code>
Function description	Disable error interrupt.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 ERRIE LL_SPI_DisableIT_ERR

LL_SPI_DisableIT_RXNE

Function name	<code>_STATIC_INLINE void LL_SPI_DisableIT_RXNE (SPI_TypeDef * SPIx)</code>
Function description	Disable Rx buffer not empty interrupt.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXNEIE LL_SPI_DisableIT_RXNE

LL_SPI_DisableIT_TXE

Function name	<code>_STATIC_INLINE void LL_SPI_DisableIT_TXE (SPI_TypeDef * SPIx)</code>
Function description	Disable Tx buffer empty interrupt.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 TXEIE LL_SPI_DisableIT_TXE

LL_SPI_IsEnabledIT_ERR

Function name	<code>_STATIC_INLINE uint32_t LL_SPI_IsEnabledIT_ERR (SPI_TypeDef * SPIx)</code>
---------------	--

Function description	Check if error interrupt is enabled.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 ERRIE LL_SPI_IsEnabledIT_ERR

LL_SPI_IsEnabledIT_RXNE

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsEnabledIT_RXNE(SPI_TypeDef * SPIx)</code>
Function description	Check if Rx buffer not empty interrupt is enabled.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 RXNEIE LL_SPI_IsEnabledIT_RXNE

LL_SPI_IsEnabledIT_TXE

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsEnabledIT_TXE(SPI_TypeDef * SPIx)</code>
Function description	Check if Tx buffer empty interrupt.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 TXEIE LL_SPI_IsEnabledIT_TXE

LL_SPI_EnableDMAReq_RX

Function name	<code>__STATIC_INLINE void LL_SPI_EnableDMAReq_RX(SPI_TypeDef * SPIx)</code>
Function description	Enable DMA Rx.
Parameters	<ul style="list-style-type: none"> • SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • CR2 RXDMAEN LL_SPI_EnableDMAReq_RX

LL_SPI_DisableDMAReq_RX

Function name	<code>__STATIC_INLINE void LL_SPI_DisableDMAReq_RX(SPI_TypeDef * SPIx)</code>
Function description	Disable DMA Rx.

Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXDMAEN LL_SPI_DisableDMAReq_RX

LL_SPI_IsEnabledDMAReq_RX

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsEnabledDMAReq_RX(SPI_TypeDef * SPIx)</code>
Function description	Check if DMA Rx is enabled.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 RXDMAEN LL_SPI_IsEnabledDMAReq_RX

LL_SPI_EnableDMAReq_TX

Function name	<code>__STATIC_INLINE void LL_SPI_EnableDMAReq_TX(SPI_TypeDef * SPIx)</code>
Function description	Enable DMA Tx.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 TXDMAEN LL_SPI_EnableDMAReq_TX

LL_SPI_DisableDMAReq_TX

Function name	<code>__STATIC_INLINE void LL_SPI_DisableDMAReq_TX(SPI_TypeDef * SPIx)</code>
Function description	Disable DMA Tx.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 TXDMAEN LL_SPI_DisableDMAReq_TX

LL_SPI_IsEnabledDMAReq_TX

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_IsEnabledDMAReq_TX(SPI_TypeDef * SPIx)</code>
Function description	Check if DMA Tx is enabled.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance

Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR2 TXDMAEN LL_SPI_IsEnabledDMAReq_TX

LL_SPI_DMA_GetRegAddr

Function name	<code>__STATIC_INLINE uint32_t LL_SPI_DMA_GetRegAddr (SPI_TypeDef * SPIx)</code>
Function description	Get the data register address used for DMA transfer.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> Address: of data register
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DR DR LL_SPI_DMA_GetRegAddr

LL_SPI_ReceiveData8

Function name	<code>__STATIC_INLINE uint8_t LL_SPI_ReceiveData8 (SPI_TypeDef * SPIx)</code>
Function description	Read 8-Bits in the data register.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> RxData: Value between Min_Data=0x00 and Max_Data=0xFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DR DR LL_SPI_ReceiveData8

LL_SPI_ReceiveData16

Function name	<code>__STATIC_INLINE uint16_t LL_SPI_ReceiveData16 (SPI_TypeDef * SPIx)</code>
Function description	Read 16-Bits in the data register.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance
Return values	<ul style="list-style-type: none"> RxData: Value between Min_Data=0x00 and Max_Data=0xFFFF
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> DR DR LL_SPI_ReceiveData16

LL_SPI_TransmitData8

Function name	<code>__STATIC_INLINE void LL_SPI_TransmitData8 (SPI_TypeDef * SPIx, uint8_t TxData)</code>
Function description	Write 8-Bits in the data register.
Parameters	<ul style="list-style-type: none"> SPIx: SPI Instance

- **TxData:** Value between Min_Data=0x00 and Max_Data=0xFF

Return values • **None:**

Reference Manual to
LL API cross
reference:
DR DR LL_SPI_TransmitData8

LL_SPI_TransmitData16

Function name **_STATIC_INLINE void LL_SPI_TransmitData16 (SPI_TypeDef * SPIx, uint16_t TxData)**

Function description Write 16-Bits in the data register.

- Parameters
- **SPIx:** SPI Instance
 - **TxData:** Value between Min_Data=0x00 and Max_Data=0xFFFF

Return values • **None:**

Reference Manual to
LL API cross
reference:
DR DR LL_SPI_TransmitData16

LL_SPI_DelInit

Function name **ErrorStatus LL_SPI_DelInit (SPI_TypeDef * SPIx)**

Function description De-initialize the SPI registers to their default reset values.

- Parameters
- **SPIx:** SPI Instance

- Return values
- **An:** ErrorStatus enumeration value:
 - SUCCESS: SPI registers are de-initialized
 - ERROR: SPI registers are not de-initialized

LL_SPI_Init

Function name **ErrorStatus LL_SPI_Init (SPI_TypeDef * SPIx,
LL_SPI_InitTypeDef * SPI_InitStruct)**

Function description Initialize the SPI registers according to the specified parameters in SPI_InitStruct.

- Parameters
- **SPIx:** SPI Instance
 - **SPI_InitStruct:** pointer to a LL_SPI_InitTypeDef structure

- Return values
- **An:** ErrorStatus enumeration value. (Return always SUCCESS)

- Notes
- As some bits in SPI configuration registers can only be written when the SPI is disabled (SPI_CR1_SPE bit =0), SPI IP should be in disabled state prior calling this function. Otherwise, ERROR result will be returned.

LL_SPI_StructInit

Function name **void LL_SPI_StructInit (LL_SPI_InitTypeDef * SPI_InitStruct)**

Function description	Set each LL_SPI_InitTypeDef field to default value.
Parameters	<ul style="list-style-type: none"> SPI_InitStruct: pointer to a LL_SPI_InitTypeDef structure whose fields will be set to default values.
Return values	<ul style="list-style-type: none"> None:

64.3 SPI Firmware driver defines

64.3.1 SPI

Baud Rate Prescaler

LL_SPI_BAUDRATEPRESCALER_DIV2	BaudRate control equal to fPCLK/2
LL_SPI_BAUDRATEPRESCALER_DIV4	BaudRate control equal to fPCLK/4
LL_SPI_BAUDRATEPRESCALER_DIV8	BaudRate control equal to fPCLK/8
LL_SPI_BAUDRATEPRESCALER_DIV16	BaudRate control equal to fPCLK/16
LL_SPI_BAUDRATEPRESCALER_DIV32	BaudRate control equal to fPCLK/32
LL_SPI_BAUDRATEPRESCALER_DIV64	BaudRate control equal to fPCLK/64
LL_SPI_BAUDRATEPRESCALER_DIV128	BaudRate control equal to fPCLK/128
LL_SPI_BAUDRATEPRESCALER_DIV256	BaudRate control equal to fPCLK/256
LL_SPI_BAUDRATEPRESCALER_DIV2	BaudRate control equal to fPCLK/2
LL_SPI_BAUDRATEPRESCALER_DIV4	BaudRate control equal to fPCLK/4
LL_SPI_BAUDRATEPRESCALER_DIV8	BaudRate control equal to fPCLK/8
LL_SPI_BAUDRATEPRESCALER_DIV16	BaudRate control equal to fPCLK/16
LL_SPI_BAUDRATEPRESCALER_DIV32	BaudRate control equal to fPCLK/32
LL_SPI_BAUDRATEPRESCALER_DIV64	BaudRate control equal to fPCLK/64
LL_SPI_BAUDRATEPRESCALER_DIV128	BaudRate control equal to fPCLK/128
LL_SPI_BAUDRATEPRESCALER_DIV256	BaudRate control equal to fPCLK/256
LL_SPI_BAUDRATEPRESCALER_DIV2	BaudRate control equal to fPCLK/2
LL_SPI_BAUDRATEPRESCALER_DIV4	BaudRate control equal to fPCLK/4
LL_SPI_BAUDRATEPRESCALER_DIV8	BaudRate control equal to fPCLK/8
LL_SPI_BAUDRATEPRESCALER_DIV16	BaudRate control equal to fPCLK/16
LL_SPI_BAUDRATEPRESCALER_DIV32	BaudRate control equal to fPCLK/32
LL_SPI_BAUDRATEPRESCALER_DIV64	BaudRate control equal to fPCLK/64
LL_SPI_BAUDRATEPRESCALER_DIV128	BaudRate control equal to fPCLK/128
LL_SPI_BAUDRATEPRESCALER_DIV256	BaudRate control equal to fPCLK/256
LL_SPI_BAUDRATEPRESCALER_DIV2	BaudRate control equal to fPCLK/2
LL_SPI_BAUDRATEPRESCALER_DIV4	BaudRate control equal to fPCLK/4
LL_SPI_BAUDRATEPRESCALER_DIV8	BaudRate control equal to fPCLK/8
LL_SPI_BAUDRATEPRESCALER_DIV16	BaudRate control equal to fPCLK/16

LL_SPI_BAUDRATEPRESCALER_DIV32	BaudRate control equal to fPCLK/32
LL_SPI_BAUDRATEPRESCALER_DIV64	BaudRate control equal to fPCLK/64
LL_SPI_BAUDRATEPRESCALER_DIV128	BaudRate control equal to fPCLK/128
LL_SPI_BAUDRATEPRESCALER_DIV256	BaudRate control equal to fPCLK/256

Transmission Bit Order

LL_SPI_LSB_FIRST	Data is transmitted/received with the LSB first
LL_SPI_MSB_FIRST	Data is transmitted/received with the MSB first
LL_SPI_LSB_FIRST	Data is transmitted/received with the LSB first
LL_SPI_MSB_FIRST	Data is transmitted/received with the MSB first
LL_SPI_LSB_FIRST	Data is transmitted/received with the LSB first
LL_SPI_MSB_FIRST	Data is transmitted/received with the MSB first
LL_SPI_LSB_FIRST	Data is transmitted/received with the LSB first
LL_SPI_MSB_FIRST	Data is transmitted/received with the MSB first

CRC Calculation

LL_SPI_CRCCALCULATION_DISABLE	CRC calculation disabled
LL_SPI_CRCCALCULATION_ENABLE	CRC calculation enabled
LL_SPI_CRCCALCULATION_DISABLE	CRC calculation disabled
LL_SPI_CRCCALCULATION_ENABLE	CRC calculation enabled
LL_SPI_CRCCALCULATION_DISABLE	CRC calculation disabled
LL_SPI_CRCCALCULATION_ENABLE	CRC calculation enabled
LL_SPI_CRCCALCULATION_DISABLE	CRC calculation disabled
LL_SPI_CRCCALCULATION_ENABLE	CRC calculation enabled

Datawidth

LL_SPI_DATAWIDTH_8BIT	Data length for SPI transfer: 8 bits
LL_SPI_DATAWIDTH_16BIT	Data length for SPI transfer: 16 bits
LL_SPI_DATAWIDTH_8BIT	Data length for SPI transfer: 8 bits
LL_SPI_DATAWIDTH_16BIT	Data length for SPI transfer: 16 bits
LL_SPI_DATAWIDTH_8BIT	Data length for SPI transfer: 8 bits
LL_SPI_DATAWIDTH_16BIT	Data length for SPI transfer: 16 bits
LL_SPI_DATAWIDTH_8BIT	Data length for SPI transfer: 8 bits
LL_SPI_DATAWIDTH_16BIT	Data length for SPI transfer: 16 bits

Get Flags Defines

LL_SPI_SR_RXNE	Rx buffer not empty flag
LL_SPI_SR_TXE	Tx buffer empty flag
LL_SPI_SR_BSY	Busy flag
LL_SPI_SR_CRCERR	CRC error flag

LL_SPI_SR_MODF	Mode fault flag
LL_SPI_SR_OVR	Overrun flag
LL_SPI_SR_FRE	TI mode frame format error flag
LL_SPI_SR_RXNE	Rx buffer not empty flag
LL_SPI_SR_TXE	Tx buffer empty flag
LL_SPI_SR_BSY	Busy flag
LL_SPI_SR_CRCERR	CRC error flag
LL_SPI_SR_MODF	Mode fault flag
LL_SPI_SR_OVR	Overrun flag
LL_SPI_SR_FRE	TI mode frame format error flag
LL_SPI_SR_RXNE	Rx buffer not empty flag
LL_SPI_SR_TXE	Tx buffer empty flag
LL_SPI_SR_BSY	Busy flag
LL_SPI_SR_CRCERR	CRC error flag
LL_SPI_SR_MODF	Mode fault flag
LL_SPI_SR_OVR	Overrun flag
LL_SPI_SR_FRE	TI mode frame format error flag
LL_SPI_SR_RXNE	Rx buffer not empty flag
LL_SPI_SR_TXE	Tx buffer empty flag
LL_SPI_SR_BSY	Busy flag
LL_SPI_SR_CRCERR	CRC error flag
LL_SPI_SR_MODF	Mode fault flag
LL_SPI_SR_OVR	Overrun flag
LL_SPI_SR_FRE	TI mode frame format error flag

IT Defines

LL_SPI_CR2_RXNEIE	Rx buffer not empty interrupt enable
LL_SPI_CR2_TXEIE	Tx buffer empty interrupt enable
LL_SPI_CR2_ERRIE	Error interrupt enable
LL_SPI_CR2_RXNEIE	Rx buffer not empty interrupt enable
LL_SPI_CR2_TXEIE	Tx buffer empty interrupt enable
LL_SPI_CR2_ERRIE	Error interrupt enable
LL_SPI_CR2_RXNEIE	Rx buffer not empty interrupt enable
LL_SPI_CR2_TXEIE	Tx buffer empty interrupt enable
LL_SPI_CR2_ERRIE	Error interrupt enable
LL_SPI_CR2_RXNEIE	Rx buffer not empty interrupt enable
LL_SPI_CR2_TXEIE	Tx buffer empty interrupt enable

`LL_SPI_CR2_ERRIE` Error interrupt enable

Operation Mode

`LL_SPI_MODE_MASTER` Master configuration

`LL_SPI_MODE_SLAVE` Slave configuration

Slave Select Pin Mode

`LL_SPI_NSS_SOFT` NSS managed internally. NSS pin not used and free

`LL_SPI_NSS_HARD_INPUT` NSS pin used in Input. Only used in Master mode

`LL_SPI_NSS_HARD_OUTPUT` NSS pin used in Output. Only used in Slave mode as chip select

`LL_SPI_NSS_SOFT` NSS managed internally. NSS pin not used and free

`LL_SPI_NSS_HARD_INPUT` NSS pin used in Input. Only used in Master mode

`LL_SPI_NSS_HARD_OUTPUT` NSS pin used in Output. Only used in Slave mode as chip select

`LL_SPI_NSS_SOFT` NSS managed internally. NSS pin not used and free

`LL_SPI_NSS_HARD_INPUT` NSS pin used in Input. Only used in Master mode

`LL_SPI_NSS_HARD_OUTPUT` NSS pin used in Output. Only used in Slave mode as chip select

`LL_SPI_NSS_SOFT` NSS managed internally. NSS pin not used and free

`LL_SPI_NSS_HARD_INPUT` NSS pin used in Input. Only used in Master mode

`LL_SPI_NSS_HARD_OUTPUT` NSS pin used in Output. Only used in Slave mode as chip select

Clock Phase

`LL_SPI_PHASE_1EDGE` First clock transition is the first data capture edge

`LL_SPI_PHASE_2EDGE` Second clock transition is the first data capture edge

`LL_SPI_PHASE_1EDGE` First clock transition is the first data capture edge

`LL_SPI_PHASE_2EDGE` Second clock transition is the first data capture edge

`LL_SPI_PHASE_1EDGE` First clock transition is the first data capture edge

`LL_SPI_PHASE_2EDGE` Second clock transition is the first data capture edge

`LL_SPI_PHASE_1EDGE` First clock transition is the first data capture edge

`LL_SPI_PHASE_2EDGE` Second clock transition is the first data capture edge

Clock Polarity

`LL_SPI_POLARITY_LOW` Clock to 0 when idle

LL_SPI_POLARITY_HIGH	Clock to 1 when idle
LL_SPI_POLARITY_LOW	Clock to 0 when idle
LL_SPI_POLARITY_HIGH	Clock to 1 when idle
LL_SPI_POLARITY_LOW	Clock to 0 when idle
LL_SPI_POLARITY_HIGH	Clock to 1 when idle
LL_SPI_POLARITY_LOW	Clock to 0 when idle
LL_SPI_POLARITY_HIGH	Clock to 1 when idle

Serial Protocol

LL_SPI_PROTOCOL_MOTOROLA	Motorola mode. Used as default value
LL_SPI_PROTOCOL_TI	TI mode
LL_SPI_PROTOCOL_MOTOROLA	Motorola mode. Used as default value
LL_SPI_PROTOCOL_TI	TI mode
LL_SPI_PROTOCOL_MOTOROLA	Motorola mode. Used as default value
LL_SPI_PROTOCOL_TI	TI mode
LL_SPI_PROTOCOL_MOTOROLA	Motorola mode. Used as default value
LL_SPI_PROTOCOL_TI	TI mode

Transfer Mode

LL_SPI_FULL_DUPLEX	Full-Duplex mode. Rx and Tx transfer on 2 lines
LL_SPI_SIMPLEX_RX	Simplex Rx mode. Rx transfer only on 1 line
LL_SPI_HALF_DUPLEX_RX	Half-Duplex Rx mode. Rx transfer on 1 line
LL_SPI_HALF_DUPLEX_TX	Half-Duplex Tx mode. Tx transfer on 1 line
LL_SPI_FULL_DUPLEX	Full-Duplex mode. Rx and Tx transfer on 2 lines
LL_SPI_SIMPLEX_RX	Simplex Rx mode. Rx transfer only on 1 line
LL_SPI_HALF_DUPLEX_RX	Half-Duplex Rx mode. Rx transfer on 1 line
LL_SPI_HALF_DUPLEX_TX	Half-Duplex Tx mode. Tx transfer on 1 line
LL_SPI_FULL_DUPLEX	Full-Duplex mode. Rx and Tx transfer on 2 lines
LL_SPI_SIMPLEX_RX	Simplex Rx mode. Rx transfer only on 1 line
LL_SPI_HALF_DUPLEX_RX	Half-Duplex Rx mode. Rx transfer on 1 line
LL_SPI_HALF_DUPLEX_TX	Half-Duplex Tx mode. Tx transfer on 1 line
LL_SPI_FULL_DUPLEX	Full-Duplex mode. Rx and Tx transfer on 2 lines
LL_SPI_SIMPLEX_RX	Simplex Rx mode. Rx transfer only on 1 line
LL_SPI_HALF_DUPLEX_RX	Half-Duplex Rx mode. Rx transfer on 1 line
LL_SPI_HALF_DUPLEX_TX	Half-Duplex Tx mode. Tx transfer on 1 line

Common Write and read registers Macros

- | | |
|-----------------|--|
| LL_SPI_WriteReg | Description: |
| | <ul style="list-style-type: none"> • Write a value in SPI register. |

Parameters:

- `__INSTANCE__`: SPI Instance
- `__REG__`: Register to be written
- `__VALUE__`: Value to be written in the register

Return value:

- None

`LL_SPI_ReadReg`

Description:

- Read a value in SPI register.

Parameters:

- `__INSTANCE__`: SPI Instance
- `__REG__`: Register to be read

Return value:

- Register: value

`LL_SPI_WriteReg`

Description:

- Write a value in SPI register.

Parameters:

- `__INSTANCE__`: SPI Instance
- `__REG__`: Register to be written
- `__VALUE__`: Value to be written in the register

Return value:

- None

`LL_SPI_ReadReg`

Description:

- Read a value in SPI register.

Parameters:

- `__INSTANCE__`: SPI Instance
- `__REG__`: Register to be read

Return value:

- Register: value

`LL_SPI_WriteReg`

Description:

- Write a value in SPI register.

Parameters:

- `__INSTANCE__`: SPI Instance
- `__REG__`: Register to be written
- `__VALUE__`: Value to be written in the register

Return value:

- None

`LL_SPI_ReadReg`

Description:

- Read a value in SPI register.

Parameters:

- __INSTANCE__: SPI Instance
- __REG__: Register to be read

Return value:

- Register: value

[LL_SPI_WriteReg](#)

Description:

- Write a value in SPI register.

Parameters:

- __INSTANCE__: SPI Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

[LL_SPI_ReadReg](#)

Description:

- Read a value in SPI register.

Parameters:

- __INSTANCE__: SPI Instance
- __REG__: Register to be read

Return value:

- Register: value

65 LL SYSTEM Generic Driver

65.1 SYSTEM Firmware driver API description

65.1.1 Detailed description of functions

LL_SYSCFG_SetRemapMemory

Function name	<code>__STATIC_INLINE void LL_SYSCFG_SetRemapMemory(uint32_t Memory)</code>
Function description	Set memory mapping at address 0x00000000.
Parameters	<ul style="list-style-type: none"> • Memory: This parameter can be one of the following values: (*) value not defined in all devices <ul style="list-style-type: none"> - <code>LL_SYSCFG_REMAP_FLASH</code> - <code>LL_SYSCFG_REMAP_SYSTEMFLASH</code> - <code>LL_SYSCFG_REMAP_SRAM</code> - <code>LL_SYSCFG_REMAP_FMC</code> (*)
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>SYSCFG_MEMRMP MEM_MODE</code> • <code>LL_SYSCFG_SetRemapMemory</code>

LL_SYSCFG_GetRemapMemory

Function name	<code>__STATIC_INLINE uint32_t LL_SYSCFG_GetRemapMemory(void)</code>
Function description	Get memory mapping at address 0x00000000.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - <code>LL_SYSCFG_REMAP_FLASH</code> - <code>LL_SYSCFG_REMAP_SYSTEMFLASH</code> - <code>LL_SYSCFG_REMAP_SRAM</code> - <code>LL_SYSCFG_REMAP_FMC</code> (*)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>SYSCFG_MEMRMP MEM_MODE</code> • <code>LL_SYSCFG_GetRemapMemory</code>

LL_SYSCFG_GetBootMode

Function name	<code>__STATIC_INLINE uint32_t LL_SYSCFG_GetBootMode(void)</code>
Function description	Return the boot mode as configured by user.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - <code>LL_SYSCFG_BOOTMODE_FLASH</code> - <code>LL_SYSCFG_BOOTMODE_SYSTEMFLASH</code> - <code>LL_SYSCFG_BOOTMODE_FSMC</code> (*)

- Reference Manual to LL API cross reference:
- LL_SYSCFG_BOOTMODE_SRAM
 - SYSCFG_MEMRMP BOOT_MODE
LL_SYSCFG_GetBootMode

LL_SYSCFG_EnableUSBPullUp

- Function name **`__STATIC_INLINE void LL_SYSCFG_EnableUSBPullUp (void)`**
- Function description Enable internal pull-up on USB DP line.
- Return values • **None:**
- Reference Manual to LL API cross reference:
- SYSCFG_PMC USB_PU LL_SYSCFG_EnableUSBPullUp

LL_SYSCFG_DisableUSBPullUp

- Function name **`__STATIC_INLINE void LL_SYSCFG_DisableUSBPullUp (void)`**
- Function description Disable internal pull-up on USB DP line.
- Return values • **None:**
- Reference Manual to LL API cross reference:
- SYSCFG_PMC USB_PU LL_SYSCFG_DisableUSBPullUp

LL_SYSCFG_EnableLCDCapacitanceConnection

- Function name **`__STATIC_INLINE void LL_SYSCFG_EnableLCDCapacitanceConnection (uint32_t Pin)`**
- Function description Enable decoupling capacitance connection.
- Parameters • **Pin:** This parameter can be a combination of the following values:
 - LL_SYSCFG_LCDCAPA_PB2
 - LL_SYSCFG_LCDCAPA_PB12
 - LL_SYSCFG_LCDCAPA_PB0
 - LL_SYSCFG_LCDCAPA_PE11
 - LL_SYSCFG_LCDCAPA_PE12
- Return values • **None:**
- Reference Manual to LL API cross reference:
- SYSCFG_PMC LCD_CAPA
LL_SYSCFG_EnableLCDCapacitanceConnection

LL_SYSCFG_DisableLCDCapacitanceConnection

- Function name **`__STATIC_INLINE void LL_SYSCFG_DisableLCDCapacitanceConnection (uint32_t Pin)`**

Function description	Disable decoupling capacitance connection.
Parameters	<ul style="list-style-type: none"> Pin: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_SYSCFG_LCDCAPA_PB2 - LL_SYSCFG_LCDCAPA_PB12 - LL_SYSCFG_LCDCAPA_PB0 - LL_SYSCFG_LCDCAPA_PE11 - LL_SYSCFG_LCDCAPA_PE12
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SYSCFG_PMC_LCD_CAPA LL_SYSCFG_DisableLCDCapacitanceConnection

LL_SYSCFG_SetEXTISource

Function name	<code>_STATIC_INLINE void LL_SYSCFG_SetEXTISource (uint32_t Port, uint32_t Line)</code>
Function description	Configure source input for the EXTI external interrupt.
Parameters	<ul style="list-style-type: none"> Port: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_SYSCFG_EXTI_PORTA - LL_SYSCFG_EXTI_PORTB - LL_SYSCFG_EXTI_PORTC - LL_SYSCFG_EXTI_PORTD - LL_SYSCFG_EXTI PORTE (*) - LL_SYSCFG_EXTI_PORTF (*) - LL_SYSCFG_EXTI_PORTG (*) - LL_SYSCFG_EXTI_PORTH Line: This parameter can be one of the following values: <ul style="list-style-type: none"> - LL_SYSCFG_EXTI_LINE0 - LL_SYSCFG_EXTI_LINE1 - LL_SYSCFG_EXTI_LINE2 - LL_SYSCFG_EXTI_LINE3 - LL_SYSCFG_EXTI_LINE4 - LL_SYSCFG_EXTI_LINE5 - LL_SYSCFG_EXTI_LINE6 - LL_SYSCFG_EXTI_LINE7 - LL_SYSCFG_EXTI_LINE8 - LL_SYSCFG_EXTI_LINE9 - LL_SYSCFG_EXTI_LINE10 - LL_SYSCFG_EXTI_LINE11 - LL_SYSCFG_EXTI_LINE12 - LL_SYSCFG_EXTI_LINE13 - LL_SYSCFG_EXTI_LINE14 - LL_SYSCFG_EXTI_LINE15
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> SYSCFG_EXTICR1 EXTI0 LL_SYSCFG_SetEXTISource SYSCFG_EXTICR1 EXTI1 LL_SYSCFG_SetEXTISource SYSCFG_EXTICR1 EXTI2 LL_SYSCFG_SetEXTISource

- SYSCFG_EXTICR4 EXTI7 LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTI8 LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTI9 LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTI10 LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTI11 LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTI12 LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTI13 LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTI14 LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTI15 LL_SYSCFG_SetEXTISource

LL_SYSCFG_GetEXTISource

Function name **`__STATIC_INLINE uint32_t LL_SYSCFG_GetEXTISource(uint32_t Line)`**

Function description Get the configured defined for specific EXTI Line.

Parameters • **Line:** This parameter can be one of the following values:

- LL_SYSCFG_EXTI_LINE0
- LL_SYSCFG_EXTI_LINE1
- LL_SYSCFG_EXTI_LINE2
- LL_SYSCFG_EXTI_LINE3
- LL_SYSCFG_EXTI_LINE4
- LL_SYSCFG_EXTI_LINE5
- LL_SYSCFG_EXTI_LINE6
- LL_SYSCFG_EXTI_LINE7
- LL_SYSCFG_EXTI_LINE8
- LL_SYSCFG_EXTI_LINE9
- LL_SYSCFG_EXTI_LINE10
- LL_SYSCFG_EXTI_LINE11
- LL_SYSCFG_EXTI_LINE12
- LL_SYSCFG_EXTI_LINE13
- LL_SYSCFG_EXTI_LINE14
- LL_SYSCFG_EXTI_LINE15

Return values • **Returned:** value can be one of the following values: (*)
value not defined in all devices.

- LL_SYSCFG_EXTI_PORTA
- LL_SYSCFG_EXTI_PORTB
- LL_SYSCFG_EXTI_PORTC
- LL_SYSCFG_EXTI_PORTD
- LL_SYSCFG_EXTI_PORTE (*)
- LL_SYSCFG_EXTI_PORTF (*)
- LL_SYSCFG_EXTI_PORTG (*)
- LL_SYSCFG_EXTI_PORTH

Reference Manual to
LL API cross
reference:

- SYSCFG_EXTICR1 EXTI0 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI1 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI2 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI3 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI4 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI5 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI6 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI7 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI8 LL_SYSCFG_GetEXTISource

- SYSCFG_EXTICR1 EXTI9 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI10 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI11 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI12 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI13 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI14 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR1 EXTI15 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI0 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI1 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI2 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI3 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI4 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI5 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI6 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI7 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI8 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI9 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI10 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI11 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI12 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI13 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI14 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTI15 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI0 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI1 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI2 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI3 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI4 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI5 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI6 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI7 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI8 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI9 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI10 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI11 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI12 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI13 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI14 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTI15 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI0 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI1 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI2 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI3 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI4 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI5 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI6 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI7 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI8 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI9 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI10 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI11 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI12 LL_SYSCFG_GetEXTISource

- SYSCFG_EXTICR4 EXTI13 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI14 LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTI15 LL_SYSCFG_GetEXTISource

LL_DBGMCU_GetDeviceID

Function name	<code>__STATIC_INLINE uint32_t LL_DBGMCU_GetDeviceID (void)</code>
Function description	Return the device identifier.
Return values	<ul style="list-style-type: none"> • Values: between Min_Data=0x00 and Max_Data=0xFFFF
Notes	<ul style="list-style-type: none"> • 0x416: Cat.1 device 0x429: Cat.2 device 0x427: Cat.3 device 0x436: Cat.4 device or Cat.3 device(1) 0x437: Cat.5 device
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DBGMCU_IDCODE DEV_ID LL_DBGMCU_GetDeviceID

LL_DBGMCU_GetRevisionID

Function name	<code>__STATIC_INLINE uint32_t LL_DBGMCU_GetRevisionID (void)</code>
Function description	Return the device revision identifier.
Return values	<ul style="list-style-type: none"> • Values: between Min_Data=0x00 and Max_Data=0xFFFF
Notes	<ul style="list-style-type: none"> • This field indicates the revision of the device. For example, it is read as Cat.1 RevA -> 0x1000, Cat.2 Rev Z -> 0x1018...
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DBGMCU_IDCODE REV_ID LL_DBGMCU_GetRevisionID

LL_DBGMCU_EnableDBGSleepMode

Function name	<code>__STATIC_INLINE void LL_DBGMCU_EnableDBGSleepMode (void)</code>
Function description	Enable the Debug Module during SLEEP mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DBGMCU_CR DBG_SLEEP LL_DBGMCU_EnableDBGSleepMode

LL_DBGMCU_DisableDBGSleepMode

Function name	<code>__STATIC_INLINE void LL_DBGMCU_DisableDBGSleepMode (void)</code>
Function description	Disable the Debug Module during SLEEP mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DBGMCU_CR DBG_SLEEP LL_DBGMCU_DisableDBGSleepMode

LL_DBGMCU_EnableDBGStopMode

Function name	<code>__STATIC_INLINE void LL_DBGMCU_EnableDBGStopMode(void)</code>
Function description	Enable the Debug Module during STOP mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>DBGMCU_CR DBG_STOP</code> • <code>LL_DBGMCU_EnableDBGStopMode</code>

LL_DBGMCU_DisableDBGStopMode

Function name	<code>__STATIC_INLINE void LL_DBGMCU_DisableDBGStopMode(void)</code>
Function description	Disable the Debug Module during STOP mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>DBGMCU_CR DBG_STOP</code> • <code>LL_DBGMCU_DisableDBGStopMode</code>

LL_DBGMCU_EnableDBGStandbyMode

Function name	<code>__STATIC_INLINE void LL_DBGMCU_EnableDBGStandbyMode(void)</code>
Function description	Enable the Debug Module during STANDBY mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>DBGMCU_CR DBG_STANDBY</code> • <code>LL_DBGMCU_EnableDBGStandbyMode</code>

LL_DBGMCU_DisableDBGStandbyMode

Function name	<code>__STATIC_INLINE void LL_DBGMCU_DisableDBGStandbyMode(void)</code>
Function description	Disable the Debug Module during STANDBY mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>DBGMCU_CR DBG_STANDBY</code> • <code>LL_DBGMCU_DisableDBGStandbyMode</code>

LL_DBGMCU_SetTracePinAssignment

Function name	<code>__STATIC_INLINE void LL_DBGMCU_SetTracePinAssignment(uint32_t PinAssignment)</code>
Function description	Set Trace pin assignment control.
Parameters	<ul style="list-style-type: none"> • PinAssignment: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_DBGMCU_TRACE_NONE</code>

	<ul style="list-style-type: none"> - LL_DBGMCU_TRACE_ASYNCH - LL_DBGMCU_TRACE_SYNCH_SIZE1 - LL_DBGMCU_TRACE_SYNCH_SIZE2 - LL_DBGMCU_TRACE_SYNCH_SIZE4
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DBGMCU_CR TRACE_IOEN LL_DBGMCU_SetTracePinAssignment • DBGMCU_CR TRACE_MODE LL_DBGMCU_SetTracePinAssignment

LL_DBGMCU_GetTracePinAssignment

Function name	<code>__STATIC_INLINE uint32_t LL_DBGMCU_GetTracePinAssignment (void)</code>
Function description	Get Trace pin assignment control.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> - LL_DBGMCU_TRACE_NONE - LL_DBGMCU_TRACE_ASYNCH - LL_DBGMCU_TRACE_SYNCH_SIZE1 - LL_DBGMCU_TRACE_SYNCH_SIZE2 - LL_DBGMCU_TRACE_SYNCH_SIZE4
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • DBGMCU_CR TRACE_IOEN LL_DBGMCU_SetTracePinAssignment • DBGMCU_CR TRACE_MODE LL_DBGMCU_SetTracePinAssignment

LL_DBGMCU_APB1_GRP1_FreezePeriph

Function name	<code>__STATIC_INLINE void LL_DBGMCU_APB1_GRP1_FreezePeriph (uint32_t Periph)</code>
Function description	Freeze APB1 peripherals (group1 peripherals)
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_DBGMCU_APB1_GRP1_TIM2_STOP - LL_DBGMCU_APB1_GRP1_TIM3_STOP - LL_DBGMCU_APB1_GRP1_TIM4_STOP - LL_DBGMCU_APB1_GRP1_TIM5_STOP (*) - LL_DBGMCU_APB1_GRP1_TIM6_STOP - LL_DBGMCU_APB1_GRP1_TIM7_STOP - LL_DBGMCU_APB1_GRP1_RTC_STOP (*) - LL_DBGMCU_APB1_GRP1_WWDG_STOP - LL_DBGMCU_APB1_GRP1_IWDG_STOP - LL_DBGMCU_APB1_GRP1_I2C1_STOP - LL_DBGMCU_APB1_GRP1_I2C2_STOP (*) value not defined in all devices.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross	<ul style="list-style-type: none"> • APB1_FZ DBG_TIM2_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph • APB1_FZ DBG_TIM3_STOP

- reference:
- LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_TIM4_STOP
 - LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_TIM5_STOP
 - LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_TIM6_STOP
 - LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_TIM7_STOP
 - LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_RTC_STOP
 - LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_WWDG_STOP
 - LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_IWDG_STOP
 - LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_I2C1_SMBUS_TIMEOUT
 - LL_DBGMCU_APB1_GRP1_FreezePeriph
 - APB1_FZ DBG_I2C2_SMBUS_TIMEOUT
 - LL_DBGMCU_APB1_GRP1_FreezePeriph

LL_DBGMCU_APB1_GRP1_UnFreezePeriph

Function name	<code>__STATIC_INLINE void LL_DBGMCU_APB1_GRP1_UnFreezePeriph (uint32_t Periph)</code>
Function description	Unfreeze APB1 peripherals (group1 peripherals)
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_DBGMCU_APB1_GRP1_TIM2_STOP - LL_DBGMCU_APB1_GRP1_TIM3_STOP - LL_DBGMCU_APB1_GRP1_TIM4_STOP - LL_DBGMCU_APB1_GRP1_TIM5_STOP (*) - LL_DBGMCU_APB1_GRP1_TIM6_STOP - LL_DBGMCU_APB1_GRP1_TIM7_STOP - LL_DBGMCU_APB1_GRP1_RTC_STOP (*) - LL_DBGMCU_APB1_GRP1_WWDG_STOP - LL_DBGMCU_APB1_GRP1_IWDG_STOP - LL_DBGMCU_APB1_GRP1_I2C1_STOP - LL_DBGMCU_APB1_GRP1_I2C2_STOP (*) value not defined in all devices.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • APB1_FZ DBG_TIM2_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph • APB1_FZ DBG_TIM3_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph • APB1_FZ DBG_TIM4_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph • APB1_FZ DBG_TIM5_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph • APB1_FZ DBG_TIM6_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph • APB1_FZ DBG_TIM7_STOP

- LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- APB1_FZ DBG_RTC_STOP
- LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- APB1_FZ DBG_WWDG_STOP
- LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- APB1_FZ DBG_IWDG_STOP
- LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- APB1_FZ DBG_I2C1_SMBUS_TIMEOUT
- LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- APB1_FZ DBG_I2C2_SMBUS_TIMEOUT
- LL_DBGMCU_APB1_GRP1_UnFreezePeriph

LL_DBGMCU_APB2_GRP1_FreezePeriph

Function name	<code>__STATIC_INLINE void LL_DBGMCU_APB2_GRP1_FreezePeriph (uint32_t Periph)</code>
Function description	Freeze APB2 peripherals.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_DBGMCU_APB2_GRP1_TIM9_STOP - LL_DBGMCU_APB2_GRP1_TIM10_STOP - LL_DBGMCU_APB2_GRP1_TIM11_STOP
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • APB2_FZ DBG_TIM9_STOP • LL_DBGMCU_APB2_GRP1_FreezePeriph • APB2_FZ DBG_TIM10_STOP • LL_DBGMCU_APB2_GRP1_FreezePeriph • APB2_FZ DBG_TIM11_STOP • LL_DBGMCU_APB2_GRP1_FreezePeriph

LL_DBGMCU_APB2_GRP1_UnFreezePeriph

Function name	<code>__STATIC_INLINE void LL_DBGMCU_APB2_GRP1_UnFreezePeriph (uint32_t Periph)</code>
Function description	Unfreeze APB2 peripherals.
Parameters	<ul style="list-style-type: none"> • Periph: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_DBGMCU_APB2_GRP1_TIM9_STOP - LL_DBGMCU_APB2_GRP1_TIM10_STOP - LL_DBGMCU_APB2_GRP1_TIM11_STOP
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • APB2_FZ DBG_TIM9_STOP • LL_DBGMCU_APB2_GRP1_UnFreezePeriph • APB2_FZ DBG_TIM10_STOP • LL_DBGMCU_APB2_GRP1_UnFreezePeriph • APB2_FZ DBG_TIM11_STOP • LL_DBGMCU_APB2_GRP1_UnFreezePeriph

LL_RI_SetRemapInputCapture_TIM

Function name	<code>_STATIC_INLINE void LL_RI_SetRemapInputCapture_TIM (uint32_t TIM_Select, uint32_t InputCaptureChannel, uint32_t Input)</code>
Function description	Configures the routing interface to map Input Capture x of TIMx to a selected I/O pin.
Parameters	<ul style="list-style-type: none"> • TIM_Select: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_RI_TIM_SELECT_NONE</code> - <code>LL_RI_TIM_SELECT_TIM2</code> - <code>LL_RI_TIM_SELECT_TIM3</code> - <code>LL_RI_TIM_SELECT_TIM4</code> • InputCaptureChannel: This parameter can be one of the following values: <ul style="list-style-type: none"> - <code>LL_RI_INPUTCAPTURE_1</code> - <code>LL_RI_INPUTCAPTURE_2</code> - <code>LL_RI_INPUTCAPTURE_3</code> - <code>LL_RI_INPUTCAPTURE_4</code> • Input: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - <code>LL_RI_INPUTCAPTUREROUTING_0</code> - <code>LL_RI_INPUTCAPTUREROUTING_1</code> - <code>LL_RI_INPUTCAPTUREROUTING_2</code> - <code>LL_RI_INPUTCAPTUREROUTING_3</code> - <code>LL_RI_INPUTCAPTUREROUTING_4</code> - <code>LL_RI_INPUTCAPTUREROUTING_5</code> - <code>LL_RI_INPUTCAPTUREROUTING_6</code> - <code>LL_RI_INPUTCAPTUREROUTING_7</code> - <code>LL_RI_INPUTCAPTUREROUTING_8</code> - <code>LL_RI_INPUTCAPTUREROUTING_9</code> - <code>LL_RI_INPUTCAPTUREROUTING_10</code> - <code>LL_RI_INPUTCAPTUREROUTING_11</code> - <code>LL_RI_INPUTCAPTUREROUTING_12 (*)</code> - <code>LL_RI_INPUTCAPTUREROUTING_13 (*)</code> - <code>LL_RI_INPUTCAPTUREROUTING_14 (*)</code> - <code>LL_RI_INPUTCAPTUREROUTING_15 (*)</code>
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RI_ICR IC1OS <code>LL_RI_SetRemapInputCapture_TIM</code> • RI_ICR IC2OS <code>LL_RI_SetRemapInputCapture_TIM</code> • RI_ICR IC3OS <code>LL_RI_SetRemapInputCapture_TIM</code> • RI_ICR IC4OS <code>LL_RI_SetRemapInputCapture_TIM</code> • RI_ICR TIM <code>LL_RI_SetRemapInputCapture_TIM</code> • RI_ICR IC1 <code>LL_RI_SetRemapInputCapture_TIM</code> • RI_ICR IC2 <code>LL_RI_SetRemapInputCapture_TIM</code> • RI_ICR IC3 <code>LL_RI_SetRemapInputCapture_TIM</code> • RI_ICR IC4 <code>LL_RI_SetRemapInputCapture_TIM</code>

LL_RI_DisableRemapInputCapture_TIM

Function name	<code>_STATIC_INLINE void</code>
---------------	---

LL_RI_DisableRemapInputCapture_TIM (uint32_t InputCaptureChannel)

Function description	Disable the TIM Input capture remap (select the standard AF)
Parameters	<ul style="list-style-type: none"> • InputCaptureChannel: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_RI_INPUTCAPTURE_1 - LL_RI_INPUTCAPTURE_2 - LL_RI_INPUTCAPTURE_3 - LL_RI_INPUTCAPTURE_4
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RI_ICR IC1 LL_RI_DisableRemapInputCapture_TIM • RI_ICR IC2 LL_RI_DisableRemapInputCapture_TIM • RI_ICR IC3 LL_RI_DisableRemapInputCapture_TIM • RI_ICR IC4 LL_RI_DisableRemapInputCapture_TIM

LL_RI_CloseIOSwitchLinkedToADC

Function name	STATIC_INLINE void LL_RI_CloseIOSwitchLinkedToADC (uint32_t IOswitch)
Function description	Close the routing interface Input Output switches linked to ADC.
Parameters	<ul style="list-style-type: none"> • IOswitch: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RI_IOSWITCH_CH0 - LL_RI_IOSWITCH_CH1 - LL_RI_IOSWITCH_CH2 - LL_RI_IOSWITCH_CH3 - LL_RI_IOSWITCH_CH4 - LL_RI_IOSWITCH_CH5 - LL_RI_IOSWITCH_CH6 - LL_RI_IOSWITCH_CH7 - LL_RI_IOSWITCH_CH8 - LL_RI_IOSWITCH_CH9 - LL_RI_IOSWITCH_CH10 - LL_RI_IOSWITCH_CH11 - LL_RI_IOSWITCH_CH12 - LL_RI_IOSWITCH_CH13 - LL_RI_IOSWITCH_CH14 - LL_RI_IOSWITCH_CH15 - LL_RI_IOSWITCH_CH18 - LL_RI_IOSWITCH_CH19 - LL_RI_IOSWITCH_CH20 - LL_RI_IOSWITCH_CH21 - LL_RI_IOSWITCH_CH22 - LL_RI_IOSWITCH_CH23 - LL_RI_IOSWITCH_CH24 - LL_RI_IOSWITCH_CH25 - LL_RI_IOSWITCH_VCOMP - LL_RI_IOSWITCH_CH27 (*) - LL_RI_IOSWITCH_CH28 (*) - LL_RI_IOSWITCH_CH29 (*)

- LL_RI_IOSWITCH_CH30 (*)
- LL_RI_IOSWITCH_CH31 (*)

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- RI_ASCR1 CH LL_RI_CloseIOSwitchLinkedToADC
- RI_ASCR1 VCOMP LL_RI_CloseIOSwitchLinkedToADC

LL_RI_OpenIOSwitchLinkedToADC

Function name

**`_STATIC_INLINE void LL_RI_OpenIOSwitchLinkedToADC
(uint32_t IOSwitch)`**

Function description

Open the routing interface Input Output switches linked to ADC.

Parameters

- **IOSwitch:** This parameter can be a combination of the following values: (*) value not defined in all devices.

- LL_RI_IOSWITCH_CH0
- LL_RI_IOSWITCH_CH1
- LL_RI_IOSWITCH_CH2
- LL_RI_IOSWITCH_CH3
- LL_RI_IOSWITCH_CH4
- LL_RI_IOSWITCH_CH5
- LL_RI_IOSWITCH_CH6
- LL_RI_IOSWITCH_CH7
- LL_RI_IOSWITCH_CH8
- LL_RI_IOSWITCH_CH9
- LL_RI_IOSWITCH_CH10
- LL_RI_IOSWITCH_CH11
- LL_RI_IOSWITCH_CH12
- LL_RI_IOSWITCH_CH13
- LL_RI_IOSWITCH_CH14
- LL_RI_IOSWITCH_CH15
- LL_RI_IOSWITCH_CH18
- LL_RI_IOSWITCH_CH19
- LL_RI_IOSWITCH_CH20
- LL_RI_IOSWITCH_CH21
- LL_RI_IOSWITCH_CH22
- LL_RI_IOSWITCH_CH23
- LL_RI_IOSWITCH_CH24
- LL_RI_IOSWITCH_CH25
- LL_RI_IOSWITCH_VCOMP
- LL_RI_IOSWITCH_CH27 (*)
- LL_RI_IOSWITCH_CH28 (*)
- LL_RI_IOSWITCH_CH29 (*)
- LL_RI_IOSWITCH_CH30 (*)
- LL_RI_IOSWITCH_CH31 (*)

Return values

- **None:**

Reference Manual to
LL API cross
reference:

- RI_ASCR1 CH LL_RI_OpenIOSwitchLinkedToADC
- RI_ASCR1 VCOMP LL_RI_OpenIOSwitchLinkedToADC

LL_RI_EnableSwitchControlMode

Function name	<code>__STATIC_INLINE void LL_RI_EnableSwitchControlMode (void)</code>
Function description	Enable the switch control mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RI_ASCR1 SCM LL_RI_EnableSwitchControlMode

LL_RI_DisableSwitchControlMode

Function name	<code>__STATIC_INLINE void LL_RI_DisableSwitchControlMode (void)</code>
Function description	Disable the switch control mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RI_ASCR1 SCM LL_RI_DisableSwitchControlMode

LL_RI_CloseIOSwitchNotLinkedToADC

Function name	<code>__STATIC_INLINE void LL_RI_CloseIOSwitchNotLinkedToADC (uint32_t IOswitch)</code>
Function description	Close the routing interface Input Output switches not linked to ADC.
Parameters	<ul style="list-style-type: none"> • IOswitch: This parameter can be a combination of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RI_IOSWITCH_GR10_1 - LL_RI_IOSWITCH_GR10_2 - LL_RI_IOSWITCH_GR10_3 - LL_RI_IOSWITCH_GR10_4 - LL_RI_IOSWITCH_GR6_1 - LL_RI_IOSWITCH_GR6_2 - LL_RI_IOSWITCH_GR5_1 - LL_RI_IOSWITCH_GR5_2 - LL_RI_IOSWITCH_GR5_3 - LL_RI_IOSWITCH_GR4_1 - LL_RI_IOSWITCH_GR4_2 - LL_RI_IOSWITCH_GR4_3 - LL_RI_IOSWITCH_CH0b (*) - LL_RI_IOSWITCH_CH1b (*) - LL_RI_IOSWITCH_CH2b (*) - LL_RI_IOSWITCH_CH3b (*) - LL_RI_IOSWITCH_CH6b (*) - LL_RI_IOSWITCH_CH7b (*) - LL_RI_IOSWITCH_CH8b (*) - LL_RI_IOSWITCH_CH9b (*) - LL_RI_IOSWITCH_CH10b (*) - LL_RI_IOSWITCH_CH11b (*)

- LL_RI_IOSWITCH_CH12b (*)
- LL_RI_IOSWITCH_GR6_3
- LL_RI_IOSWITCH_GR6_4

Return values

Reference Manual to
LL API cross
reference:

- **None:**
- RI_ASCR2 GR10_1 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR10_2 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR10_3 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR10_4 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR6_1 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR6_2 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR5_1 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR5_2 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR5_3 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR4_1 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR4_2 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR4_3 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR4_4 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH0b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH1b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH2b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH3b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH6b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH7b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH8b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH9b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH10b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH11b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 CH12b LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR6_3 LL_RI_CloseIOSwitchNotLinkedToADC
- RI_ASCR2 GR6_4 LL_RI_CloseIOSwitchNotLinkedToADC

LL_RI_OpenIOSwitchNotLinkedToADC

Function name

**_STATIC_INLINE void LL_RI_OpenIOSwitchNotLinkedToADC
(uint32_t IOswitch)**

Function description

Open the routing interface Input Output switches not linked to ADC.

Parameters

- **IOswitch:** This parameter can be a combination of the following values: (*) value not defined in all devices.
 - LL_RI_IOSWITCH_GR10_1
 - LL_RI_IOSWITCH_GR10_2
 - LL_RI_IOSWITCH_GR10_3
 - LL_RI_IOSWITCH_GR10_4
 - LL_RI_IOSWITCH_GR6_1
 - LL_RI_IOSWITCH_GR6_2
 - LL_RI_IOSWITCH_GR5_1
 - LL_RI_IOSWITCH_GR5_2
 - LL_RI_IOSWITCH_GR5_3
 - LL_RI_IOSWITCH_GR4_1
 - LL_RI_IOSWITCH_GR4_2
 - LL_RI_IOSWITCH_GR4_3

- LL_RI_IOSWITCH_CH0b (*)
- LL_RI_IOSWITCH_CH1b (*)
- LL_RI_IOSWITCH_CH2b (*)
- LL_RI_IOSWITCH_CH3b (*)
- LL_RI_IOSWITCH_CH6b (*)
- LL_RI_IOSWITCH_CH7b (*)
- LL_RI_IOSWITCH_CH8b (*)
- LL_RI_IOSWITCH_CH9b (*)
- LL_RI_IOSWITCH_CH10b (*)
- LL_RI_IOSWITCH_CH11b (*)
- LL_RI_IOSWITCH_CH12b (*)
- LL_RI_IOSWITCH_GR6_3
- LL_RI_IOSWITCH_GR6_4

Return values

Reference Manual to
LL API cross
reference:

- **None:**
- RI_ASCR2 GR10_1 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR10_2 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR10_3 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR10_4 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR6_1 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR6_2 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR5_1 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR5_2 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR5_3 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR4_1 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR4_2 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR4_3 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR4_4 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH0b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH1b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH2b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH3b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH6b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH7b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH8b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH9b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH10b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH11b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 CH12b LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR6_3 LL_RI_OpenIOSwitchNotLinkedToADC
- RI_ASCR2 GR6_4 LL_RI_OpenIOSwitchNotLinkedToADC

LL_RI_EnableHysteresis

Function name

_STATIC_INLINE void LL_RI_EnableHysteresis (uint32_t Port, uint32_t Pin)

Function description

Enable Hysteresis of the input schmitt trigger of the port X.

Parameters

- **Port:** This parameter can be one of the following values: (*)
value not defined in all devices.
 - LL_RI_HSYTERESIS_PORT_A
 - LL_RI_HSYTERESIS_PORT_B
 - LL_RI_HSYTERESIS_PORT_C

- LL_RI_HSYTERESIS_PORT_D
- LL_RI_HSYTERESIS_PORT_E (*)
- LL_RI_HSYTERESIS_PORT_F (*)
- LL_RI_HSYTERESIS_PORT_G (*)
- **Pin:** This parameter can be a combination of the following values:
 - LL_RI_PIN_0
 - LL_RI_PIN_1
 - LL_RI_PIN_2
 - LL_RI_PIN_3
 - LL_RI_PIN_4
 - LL_RI_PIN_5
 - LL_RI_PIN_6
 - LL_RI_PIN_7
 - LL_RI_PIN_8
 - LL_RI_PIN_9
 - LL_RI_PIN_10
 - LL_RI_PIN_11
 - LL_RI_PIN_12
 - LL_RI_PIN_13
 - LL_RI_PIN_14
 - LL_RI_PIN_15
 - LL_RI_PIN_ALL

Return values

Reference Manual to
LL API cross
reference:

- **None:**
- RI_HYSCR1 PA LL_RI_EnableHysteresis
- RI_HYSCR1 PB LL_RI_EnableHysteresis
- RI_HYSCR1 PC LL_RI_EnableHysteresis
- RI_HYSCR1 PD LL_RI_EnableHysteresis
- RI_HYSCR1 PE LL_RI_EnableHysteresis
- RI_HYSCR1 PF LL_RI_EnableHysteresis
- RI_HYSCR1 PG LL_RI_EnableHysteresis
- RI_HYSCR2 PA LL_RI_EnableHysteresis
- RI_HYSCR2 PB LL_RI_EnableHysteresis
- RI_HYSCR2 PC LL_RI_EnableHysteresis
- RI_HYSCR2 PD LL_RI_EnableHysteresis
- RI_HYSCR2 PE LL_RI_EnableHysteresis
- RI_HYSCR2 PF LL_RI_EnableHysteresis
- RI_HYSCR2 PG LL_RI_EnableHysteresis
- RI_HYSCR3 PA LL_RI_EnableHysteresis
- RI_HYSCR3 PB LL_RI_EnableHysteresis
- RI_HYSCR3 PC LL_RI_EnableHysteresis
- RI_HYSCR3 PD LL_RI_EnableHysteresis
- RI_HYSCR3 PE LL_RI_EnableHysteresis
- RI_HYSCR3 PF LL_RI_EnableHysteresis
- RI_HYSCR3 PG LL_RI_EnableHysteresis
- RI_HYSCR4 PA LL_RI_EnableHysteresis
- RI_HYSCR4 PB LL_RI_EnableHysteresis
- RI_HYSCR4 PC LL_RI_EnableHysteresis
- RI_HYSCR4 PD LL_RI_EnableHysteresis
- RI_HYSCR4 PE LL_RI_EnableHysteresis
- RI_HYSCR4 PF LL_RI_EnableHysteresis

- RI_HYSCR4 PG LL_RI_EnableHysteresis

LL_RI_DisableHysteresis

Function name	<code>__STATIC_INLINE void LL_RI_DisableHysteresis (uint32_t Port, uint32_t Pin)</code>
Function description	Disable Hysteresis of the input schmitt trigger of the port X.
Parameters	<ul style="list-style-type: none"> • Port: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> – LL_RI_HSYTERESIS_PORT_A – LL_RI_HSYTERESIS_PORT_B – LL_RI_HSYTERESIS_PORT_C – LL_RI_HSYTERESIS_PORT_D – LL_RI_HSYTERESIS_PORT_E (*) – LL_RI_HSYTERESIS_PORT_F (*) – LL_RI_HSYTERESIS_PORT_G (*) • Pin: This parameter can be a combination of the following values: <ul style="list-style-type: none"> – LL_RI_PIN_0 – LL_RI_PIN_1 – LL_RI_PIN_2 – LL_RI_PIN_3 – LL_RI_PIN_4 – LL_RI_PIN_5 – LL_RI_PIN_6 – LL_RI_PIN_7 – LL_RI_PIN_8 – LL_RI_PIN_9 – LL_RI_PIN_10 – LL_RI_PIN_11 – LL_RI_PIN_12 – LL_RI_PIN_13 – LL_RI_PIN_14 – LL_RI_PIN_15 – LL_RI_PIN_ALL
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RI_HYSCR1 PA LL_RI_DisableHysteresis • RI_HYSCR1 PB LL_RI_DisableHysteresis • RI_HYSCR1 PC LL_RI_DisableHysteresis • RI_HYSCR1 PD LL_RI_DisableHysteresis • RI_HYSCR1 PE LL_RI_DisableHysteresis • RI_HYSCR1 PF LL_RI_DisableHysteresis • RI_HYSCR1 PG LL_RI_DisableHysteresis • RI_HYSCR2 PA LL_RI_DisableHysteresis • RI_HYSCR2 PB LL_RI_DisableHysteresis • RI_HYSCR2 PC LL_RI_DisableHysteresis • RI_HYSCR2 PD LL_RI_DisableHysteresis • RI_HYSCR2 PE LL_RI_DisableHysteresis • RI_HYSCR2 PF LL_RI_DisableHysteresis • RI_HYSCR2 PG LL_RI_DisableHysteresis • RI_HYSCR3 PA LL_RI_DisableHysteresis

- RI_HYSCR3 PB LL_RI_DisableHysteresis
- RI_HYSCR3 PC LL_RI_DisableHysteresis
- RI_HYSCR3 PD LL_RI_DisableHysteresis
- RI_HYSCR3 PE LL_RI_DisableHysteresis
- RI_HYSCR3 PF LL_RI_DisableHysteresis
- RI_HYSCR3 PG LL_RI_DisableHysteresis
- RI_HYSCR4 PA LL_RI_DisableHysteresis
- RI_HYSCR4 PB LL_RI_DisableHysteresis
- RI_HYSCR4 PC LL_RI_DisableHysteresis
- RI_HYSCR4 PD LL_RI_DisableHysteresis
- RI_HYSCR4 PE LL_RI_DisableHysteresis
- RI_HYSCR4 PF LL_RI_DisableHysteresis
- RI_HYSCR4 PG LL_RI_DisableHysteresis

LL_RI_ControlSwitchByADC

Function name	<code>__STATIC_INLINE void LL_RI_ControlSwitchByADC (uint32_t Port, uint32_t Pin)</code>
Function description	Control analog switches of port X through the ADC interface or RI_ASCRx registers.
Parameters	<ul style="list-style-type: none"> • Port: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RI_PORT_A - LL_RI_PORT_B - LL_RI_PORT_C - LL_RI_PORT_F (*) - LL_RI_PORT_G (*) • Pin: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_RI_PIN_0 - LL_RI_PIN_1 - LL_RI_PIN_2 - LL_RI_PIN_3 - LL_RI_PIN_4 - LL_RI_PIN_5 - LL_RI_PIN_6 - LL_RI_PIN_7 - LL_RI_PIN_8 - LL_RI_PIN_9 - LL_RI_PIN_10 - LL_RI_PIN_11 - LL_RI_PIN_12 - LL_RI_PIN_13 - LL_RI_PIN_14 - LL_RI_PIN_15 - LL_RI_PIN_ALL
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RI_ASMR1 PA LL_RI_ControlSwitchByADC • RI_ASMR1 PB LL_RI_ControlSwitchByADC • RI_ASMR1 PC LL_RI_ControlSwitchByADC • RI_ASMR1 PF LL_RI_ControlSwitchByADC

- RI_ASMR1 PG LL_RI_ControlSwitchByADC
- RI_ASMR2 PA LL_RI_ControlSwitchByADC
- RI_ASMR2 PB LL_RI_ControlSwitchByADC
- RI_ASMR2 PC LL_RI_ControlSwitchByADC
- RI_ASMR2 PF LL_RI_ControlSwitchByADC
- RI_ASMR2 PG LL_RI_ControlSwitchByADC
- RI_ASMR3 PA LL_RI_ControlSwitchByADC
- RI_ASMR3 PB LL_RI_ControlSwitchByADC
- RI_ASMR3 PC LL_RI_ControlSwitchByADC
- RI_ASMR3 PF LL_RI_ControlSwitchByADC
- RI_ASMR3 PG LL_RI_ControlSwitchByADC
- RI_ASMR4 PA LL_RI_ControlSwitchByADC
- RI_ASMR4 PB LL_RI_ControlSwitchByADC
- RI_ASMR4 PC LL_RI_ControlSwitchByADC
- RI_ASMR4 PF LL_RI_ControlSwitchByADC
- RI_ASMR4 PG LL_RI_ControlSwitchByADC
- RI_ASMR5 PA LL_RI_ControlSwitchByADC
- RI_ASMR5 PB LL_RI_ControlSwitchByADC
- RI_ASMR5 PC LL_RI_ControlSwitchByADC
- RI_ASMR5 PF LL_RI_ControlSwitchByADC
- RI_ASMR5 PG LL_RI_ControlSwitchByADC

LL_RI_ControlSwitchByTIM

Function name	<code>__STATIC_INLINE void LL_RI_ControlSwitchByTIM (uint32_t Port, uint32_t Pin)</code>
Function description	Control analog switches of port X by the timer OC.
Parameters	<ul style="list-style-type: none"> • Port: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - LL_RI_PORT_A - LL_RI_PORT_B - LL_RI_PORT_C - LL_RI_PORT_F (*) - LL_RI_PORT_G (*) • Pin: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - LL_RI_PIN_0 - LL_RI_PIN_1 - LL_RI_PIN_2 - LL_RI_PIN_3 - LL_RI_PIN_4 - LL_RI_PIN_5 - LL_RI_PIN_6 - LL_RI_PIN_7 - LL_RI_PIN_8 - LL_RI_PIN_9 - LL_RI_PIN_10 - LL_RI_PIN_11 - LL_RI_PIN_12 - LL_RI_PIN_13 - LL_RI_PIN_14 - LL_RI_PIN_15

	– LL_RI_PIN_ALL
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • RI_ASMR1 PA LL_RI_ControlSwitchByTIM • RI_ASMR1 PB LL_RI_ControlSwitchByTIM • RI_ASMR1 PC LL_RI_ControlSwitchByTIM • RI_ASMR1 PF LL_RI_ControlSwitchByTIM • RI_ASMR1 PG LL_RI_ControlSwitchByTIM • RI_ASMR2 PA LL_RI_ControlSwitchByTIM • RI_ASMR2 PB LL_RI_ControlSwitchByTIM • RI_ASMR2 PC LL_RI_ControlSwitchByTIM • RI_ASMR2 PF LL_RI_ControlSwitchByTIM • RI_ASMR2 PG LL_RI_ControlSwitchByTIM • RI_ASMR3 PA LL_RI_ControlSwitchByTIM • RI_ASMR3 PB LL_RI_ControlSwitchByTIM • RI_ASMR3 PC LL_RI_ControlSwitchByTIM • RI_ASMR3 PF LL_RI_ControlSwitchByTIM • RI_ASMR3 PG LL_RI_ControlSwitchByTIM • RI_ASMR4 PA LL_RI_ControlSwitchByTIM • RI_ASMR4 PB LL_RI_ControlSwitchByTIM • RI_ASMR4 PC LL_RI_ControlSwitchByTIM • RI_ASMR4 PF LL_RI_ControlSwitchByTIM • RI_ASMR4 PG LL_RI_ControlSwitchByTIM • RI_ASMR5 PA LL_RI_ControlSwitchByTIM • RI_ASMR5 PB LL_RI_ControlSwitchByTIM • RI_ASMR5 PC LL_RI_ControlSwitchByTIM • RI_ASMR5 PF LL_RI_ControlSwitchByTIM • RI_ASMR5 PG LL_RI_ControlSwitchByTIM

LL_RI_MaskChannelDuringAcquisition

Function name	STATIC_INLINE void LL_RI_MaskChannelDuringAcquisition (uint32_t Port, uint32_t Pin)
Function description	Mask the input of port X during the capacitive sensing acquisition.
Parameters	<ul style="list-style-type: none"> • Port: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> – LL_RI_PORT_A – LL_RI_PORT_B – LL_RI_PORT_C – LL_RI_PORT_F (*) – LL_RI_PORT_G (*) • Pin: This parameter can be a combination of the following values: <ul style="list-style-type: none"> – LL_RI_PIN_0 – LL_RI_PIN_1 – LL_RI_PIN_2 – LL_RI_PIN_3 – LL_RI_PIN_4 – LL_RI_PIN_5 – LL_RI_PIN_6 – LL_RI_PIN_7 – LL_RI_PIN_8

- LL_RI_PIN_9
- LL_RI_PIN_10
- LL_RI_PIN_11
- LL_RI_PIN_12
- LL_RI_PIN_13
- LL_RI_PIN_14
- LL_RI_PIN_15
- LL_RI_PIN_ALL

Return values

Reference Manual to
LL API cross
reference:

- **None:**
- RI_CMRI PA LL_RI_MaskChannelDuringAcquisition
- RI_CMRI PB LL_RI_MaskChannelDuringAcquisition
- RI_CMRI PC LL_RI_MaskChannelDuringAcquisition
- RI_CMRI PF LL_RI_MaskChannelDuringAcquisition
- RI_CMRI PG LL_RI_MaskChannelDuringAcquisition
- RI_CMRI2 PA LL_RI_MaskChannelDuringAcquisition
- RI_CMRI2 PB LL_RI_MaskChannelDuringAcquisition
- RI_CMRI2 PC LL_RI_MaskChannelDuringAcquisition
- RI_CMRI2 PF LL_RI_MaskChannelDuringAcquisition
- RI_CMRI2 PG LL_RI_MaskChannelDuringAcquisition
- RI_CMRI3 PA LL_RI_MaskChannelDuringAcquisition
- RI_CMRI3 PB LL_RI_MaskChannelDuringAcquisition
- RI_CMRI3 PC LL_RI_MaskChannelDuringAcquisition
- RI_CMRI3 PF LL_RI_MaskChannelDuringAcquisition
- RI_CMRI3 PG LL_RI_MaskChannelDuringAcquisition
- RI_CMRI4 PA LL_RI_MaskChannelDuringAcquisition
- RI_CMRI4 PB LL_RI_MaskChannelDuringAcquisition
- RI_CMRI4 PC LL_RI_MaskChannelDuringAcquisition
- RI_CMRI4 PF LL_RI_MaskChannelDuringAcquisition
- RI_CMRI4 PG LL_RI_MaskChannelDuringAcquisition
- RI_CMRI5 PA LL_RI_MaskChannelDuringAcquisition
- RI_CMRI5 PB LL_RI_MaskChannelDuringAcquisition
- RI_CMRI5 PC LL_RI_MaskChannelDuringAcquisition
- RI_CMRI5 PF LL_RI_MaskChannelDuringAcquisition
- RI_CMRI5 PG LL_RI_MaskChannelDuringAcquisition

LL_RI_UnmaskChannelDuringAcquisition**Function name**

**STATIC_INLINE void
LL_RI_UnmaskChannelDuringAcquisition (uint32_t Port,
uint32_t Pin)**

Function description

Unmask the input of port X during the capacitive sensing acquisition.

Parameters

- **Port:** This parameter can be one of the following values: (*) value not defined in all devices.
 - LL_RI_PORT_A
 - LL_RI_PORT_B
 - LL_RI_PORT_C
 - LL_RI_PORT_F (*)
 - LL_RI_PORT_G (*)
- **Pin:** This parameter can be a combination of the following values:

- LL_RI_PIN_0
- LL_RI_PIN_1
- LL_RI_PIN_2
- LL_RI_PIN_3
- LL_RI_PIN_4
- LL_RI_PIN_5
- LL_RI_PIN_6
- LL_RI_PIN_7
- LL_RI_PIN_8
- LL_RI_PIN_9
- LL_RI_PIN_10
- LL_RI_PIN_11
- LL_RI_PIN_12
- LL_RI_PIN_13
- LL_RI_PIN_14
- LL_RI_PIN_15
- LL_RI_PIN_ALL

Return values

Reference Manual to
LL API cross
reference:

- **None:**
- RI_CMRI PA LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI PB LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI PC LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI PF LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI PG LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI2 PA LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI2 PB LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI2 PC LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI2 PF LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI2 PG LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI3 PA LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI3 PB LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI3 PC LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI3 PF LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI3 PG LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI4 PA LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI4 PB LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI4 PC LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI4 PF LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI4 PG LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI5 PA LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI5 PB LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI5 PC LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI5 PF LL_RI_UnmaskChannelDuringAcquisition
- RI_CMRI5 PG LL_RI_UnmaskChannelDuringAcquisition

LL_RI_IdentifyChannelIO

Function name	<code>__STATIC_INLINE void LL_RI_IdentifyChannelIO (uint32_t Port, uint32_t Pin)</code>
Function description	Identify channel for timer input capture.
Parameters	<ul style="list-style-type: none"> • Port: This parameter can be one of the following values: (*) value not defined in all devices.

- LL_RI_PORT_A
- LL_RI_PORT_B
- LL_RI_PORT_C
- LL_RI_PORT_F (*)
- LL_RI_PORT_G (*)
- **Pin:** This parameter can be a combination of the following values:
 - LL_RI_PIN_0
 - LL_RI_PIN_1
 - LL_RI_PIN_2
 - LL_RI_PIN_3
 - LL_RI_PIN_4
 - LL_RI_PIN_5
 - LL_RI_PIN_6
 - LL_RI_PIN_7
 - LL_RI_PIN_8
 - LL_RI_PIN_9
 - LL_RI_PIN_10
 - LL_RI_PIN_11
 - LL_RI_PIN_12
 - LL_RI_PIN_13
 - LL_RI_PIN_14
 - LL_RI_PIN_15
 - LL_RI_PIN_ALL

Return values

Reference Manual to
LL API cross
reference:

- **None:**
- RI_CICR1 PA LL_RI_IdentifyChannelIO
- RI_CICR1 PB LL_RI_IdentifyChannelIO
- RI_CICR1 PC LL_RI_IdentifyChannelIO
- RI_CICR1 PF LL_RI_IdentifyChannelIO
- RI_CICR1 PG LL_RI_IdentifyChannelIO
- RI_CICR2 PA LL_RI_IdentifyChannelIO
- RI_CICR2 PB LL_RI_IdentifyChannelIO
- RI_CICR2 PC LL_RI_IdentifyChannelIO
- RI_CICR2 PF LL_RI_IdentifyChannelIO
- RI_CICR2 PG LL_RI_IdentifyChannelIO
- RI_CICR3 PA LL_RI_IdentifyChannelIO
- RI_CICR3 PB LL_RI_IdentifyChannelIO
- RI_CICR3 PC LL_RI_IdentifyChannelIO
- RI_CICR3 PF LL_RI_IdentifyChannelIO
- RI_CICR3 PG LL_RI_IdentifyChannelIO
- RI_CICR4 PA LL_RI_IdentifyChannelIO
- RI_CICR4 PB LL_RI_IdentifyChannelIO
- RI_CICR4 PC LL_RI_IdentifyChannelIO
- RI_CICR4 PF LL_RI_IdentifyChannelIO
- RI_CICR4 PG LL_RI_IdentifyChannelIO
- RI_CICR5 PA LL_RI_IdentifyChannelIO
- RI_CICR5 PB LL_RI_IdentifyChannelIO
- RI_CICR5 PC LL_RI_IdentifyChannelIO
- RI_CICR5 PF LL_RI_IdentifyChannelIO
- RI_CICR5 PG LL_RI_IdentifyChannelIO

LL_RI_IdentifySamplingCapacitorIO

Function name	<code>_STATIC_INLINE void LL_RI_IdentifySamplingCapacitorIO (uint32_t Port, uint32_t Pin)</code>
Function description	Identify sampling capacitor for timer input capture.
Parameters	<ul style="list-style-type: none"> • Port: This parameter can be one of the following values: (*) value not defined in all devices. <ul style="list-style-type: none"> - <code>LL_RI_PORT_A</code> - <code>LL_RI_PORT_B</code> - <code>LL_RI_PORT_C</code> - <code>LL_RI_PORT_F (*)</code> - <code>LL_RI_PORT_G (*)</code> • Pin: This parameter can be a combination of the following values: <ul style="list-style-type: none"> - <code>LL_RI_PIN_0</code> - <code>LL_RI_PIN_1</code> - <code>LL_RI_PIN_2</code> - <code>LL_RI_PIN_3</code> - <code>LL_RI_PIN_4</code> - <code>LL_RI_PIN_5</code> - <code>LL_RI_PIN_6</code> - <code>LL_RI_PIN_7</code> - <code>LL_RI_PIN_8</code> - <code>LL_RI_PIN_9</code> - <code>LL_RI_PIN_10</code> - <code>LL_RI_PIN_11</code> - <code>LL_RI_PIN_12</code> - <code>LL_RI_PIN_13</code> - <code>LL_RI_PIN_14</code> - <code>LL_RI_PIN_15</code> - <code>LL_RI_PIN_ALL</code>
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • <code>RI_CICR1 PA LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR1 PB LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR1 PC LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR1 PF LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR1 PG LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR2 PA LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR2 PB LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR2 PC LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR2 PF LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR2 PG LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR3 PA LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR3 PB LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR3 PC LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR3 PF LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR3 PG LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR4 PA LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR4 PB LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR4 PC LL_RI_IdentifySamplingCapacitorIO</code> • <code>RI_CICR4 PF LL_RI_IdentifySamplingCapacitorIO</code>

- RI_CICR4 PG LL_RI_IdentifySamplingCapacitorIO
- RI_CICR5 PA LL_RI_IdentifySamplingCapacitorIO
- RI_CICR5 PB LL_RI_IdentifySamplingCapacitorIO
- RI_CICR5 PC LL_RI_IdentifySamplingCapacitorIO
- RI_CICR5 PF LL_RI_IdentifySamplingCapacitorIO
- RI_CICR5 PG LL_RI_IdentifySamplingCapacitorIO

LL_FLASH_SetLatency

Function name	<code>__STATIC_INLINE void LL_FLASH_SetLatency (uint32_t Latency)</code>
Function description	Set FLASH Latency.
Parameters	<ul style="list-style-type: none"> • Latency: This parameter can be one of the following values: <ul style="list-style-type: none"> – LL_FLASH_LATENCY_0 – LL_FLASH_LATENCY_1
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Latency can be modified only when ACC64 is set. (through function LL_FLASH_Enable64bitAccess)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • FLASH_ACR LATENCY LL_FLASH_SetLatency

LL_FLASH_GetLatency

Function name	<code>__STATIC_INLINE uint32_t LL_FLASH_GetLatency (void)</code>
Function description	Get FLASH Latency.
Return values	<ul style="list-style-type: none"> • Returned: value can be one of the following values: <ul style="list-style-type: none"> – LL_FLASH_LATENCY_0 – LL_FLASH_LATENCY_1
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • FLASH_ACR LATENCY LL_FLASH_GetLatency

LL_FLASH_EnablePrefetch

Function name	<code>__STATIC_INLINE void LL_FLASH_EnablePrefetch (void)</code>
Function description	Enable Prefetch.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Prefetch can be enabled only when ACC64 is set. (through function LL_FLASH_Enable64bitAccess)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • FLASH_ACR PRFTEN LL_FLASH_EnablePrefetch

LL_FLASH_DisablePrefetch

Function name	<code>__STATIC_INLINE void LL_FLASH_DisablePrefetch (void)</code>
---------------	--

Function description	Disable Prefetch.
Return values	<ul style="list-style-type: none"> None:
Notes	<ul style="list-style-type: none"> Prefetch can be disabled only when ACC64 is set. (through function LL_FLASH_Enable64bitAccess)
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> FLASH_ACR PRFTEN LL_FLASH_DisablePrefetch

LL_FLASH_IsPrefetchEnabled

Function name	<code>__STATIC_INLINE uint32_t LL_FLASH_IsPrefetchEnabled (void)</code>
Function description	Check if Prefetch buffer is enabled.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> FLASH_ACR PRFTEN LL_FLASH_IsPrefetchEnabled

LL_FLASH_Enable64bitAccess

Function name	<code>__STATIC_INLINE void LL_FLASH_Enable64bitAccess (void)</code>
Function description	Enable 64-bit access.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> FLASH_ACR ACC64 LL_FLASH_Enable64bitAccess

LL_FLASH_Disable64bitAccess

Function name	<code>__STATIC_INLINE void LL_FLASH_Disable64bitAccess (void)</code>
Function description	Disable 64-bit access.
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> FLASH_ACR ACC64 LL_FLASH_Disable64bitAccess

LL_FLASH_Is64bitAccessEnabled

Function name	<code>__STATIC_INLINE uint32_t LL_FLASH_Is64bitAccessEnabled (void)</code>
Function description	Check if 64-bit access is enabled.
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> FLASH_ACR ACC64 LL_FLASH_Is64bitAccessEnabled

LL_FLASH_EnableRunPowerDown

Function name	<code>__STATIC_INLINE void LL_FLASH_EnableRunPowerDown(void)</code>
Function description	Enable Flash Power-down mode during run mode or Low-power run mode.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Flash memory can be put in power-down mode only when the code is executed from RAM • Flash must not be accessed when power down is enabled • Flash must not be put in power-down while a program or an erase operation is on-going
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • FLASH_ACR RUN_PD LL_FLASH_EnableRunPowerDown • FLASH_PDKEYR PDKEY1 LL_FLASH_EnableRunPowerDown • FLASH_PDKEYR PDKEY2 LL_FLASH_EnableRunPowerDown

LL_FLASH_DisableRunPowerDown

Function name	<code>__STATIC_INLINE void LL_FLASH_DisableRunPowerDown(void)</code>
Function description	Disable Flash Power-down mode during run mode or Low-power run mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • FLASH_ACR RUN_PD LL_FLASH_DisableRunPowerDown • FLASH_PDKEYR PDKEY1 LL_FLASH_DisableRunPowerDown • FLASH_PDKEYR PDKEY2 LL_FLASH_DisableRunPowerDown

LL_FLASH_EnableSleepPowerDown

Function name	<code>__STATIC_INLINE void LL_FLASH_EnableSleepPowerDown(void)</code>
Function description	Enable Flash Power-down mode during Sleep or Low-power sleep mode.
Return values	<ul style="list-style-type: none"> • None:
Notes	<ul style="list-style-type: none"> • Flash must not be put in power-down while a program or an erase operation is on-going
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • FLASH_ACR SLEEP_PD LL_FLASH_EnableSleepPowerDown

LL_FLASH_DisableSleepPowerDown

Function name	<code>__STATIC_INLINE void LL_FLASH_DisableSleepPowerDown(void)</code>
---------------	--

Function description	Disable Flash Power-down mode during Sleep or Low-power sleep mode.
Return values	<ul style="list-style-type: none"> • None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> • FLASH_ACR_SLEEP_PD • LL_FLASH_DisableSleepPowerDown

65.2 SYSTEM Firmware driver defines

65.2.1 SYSTEM

DBGMCU APB1 GRP1 STOP IP

LL_DBGMCU_APB1_GRP1_TIM2_STOP	TIM2 counter stopped when core is halted
LL_DBGMCU_APB1_GRP1_TIM3_STOP	TIM3 counter stopped when core is halted
LL_DBGMCU_APB1_GRP1_TIM4_STOP	TIM4 counter stopped when core is halted
LL_DBGMCU_APB1_GRP1_TIM5_STOP	TIM5 counter stopped when core is halted
LL_DBGMCU_APB1_GRP1_TIM6_STOP	TIM6 counter stopped when core is halted
LL_DBGMCU_APB1_GRP1_TIM7_STOP	TIM7 counter stopped when core is halted
LL_DBGMCU_APB1_GRP1_RTC_STOP	RTC Counter stopped when Core is halted
LL_DBGMCU_APB1_GRP1_WWDG_STOP	Debug Window Watchdog stopped when Core is halted
LL_DBGMCU_APB1_GRP1_IWDG_STOP	Debug Independent Watchdog stopped when Core is halted
LL_DBGMCU_APB1_GRP1_I2C1_STOP	I2C1 SMBUS timeout mode stopped when Core is halted
LL_DBGMCU_APB1_GRP1_I2C2_STOP	I2C2 SMBUS timeout mode stopped when Core is halted

DBGMCU APB2 GRP1 STOP IP

LL_DBGMCU_APB2_GRP1_TIM9_STOP	TIM9 counter stopped when core is halted
LL_DBGMCU_APB2_GRP1_TIM10_STOP	TIM10 counter stopped when core is halted
LL_DBGMCU_APB2_GRP1_TIM11_STOP	TIM11 counter stopped when core is halted

SYSCFG BOOT MODE

LL_SYSCFG_BOOTMODE_FLASH	
LL_SYSCFG_BOOTMODE_SYSTEMFLASH	
LL_SYSCFG_BOOTMODE_FSMC	
LL_SYSCFG_BOOTMODE_SRAM	

SYSCFG EXTI PORT

LL_SYSCFG_EXTI_PORTA	EXTI PORT A
LL_SYSCFG_EXTI_PORTB	EXTI PORT B
LL_SYSCFG_EXTI_PORTC	EXTI PORT C

LL_SYSCFG_EXTI_PORTD	EXTI PORT D
LL_SYSCFG_EXTI PORTE	EXTI PORT E
LL_SYSCFG_EXTI_PORTF	EXTI PORT F
LL_SYSCFG_EXTI_PORTG	EXTI PORT G
LL_SYSCFG_EXTI_PORTH	EXTI PORT H

RI HYSTERESIS PORT

LL_RI_HYSYTERESIS_PORT_A	HYSYTERESIS PORT A
LL_RI_HYSYTERESIS_PORT_B	HYSYTERESIS PORT B
LL_RI_HYSYTERESIS_PORT_C	HYSYTERESIS PORT C
LL_RI_HYSYTERESIS_PORT_D	HYSYTERESIS PORT D
LL_RI_HYSYTERESIS_PORT_E	HYSYTERESIS PORT E
LL_RI_HYSYTERESIS_PORT_F	HYSYTERESIS PORT F
LL_RI_HYSYTERESIS_PORT_G	HYSYTERESIS PORT G

RI Input Capture number

LL_RI_INPUTCAPTURE_1	Input Capture 1 select output
LL_RI_INPUTCAPTURE_2	Input Capture 2 select output
LL_RI_INPUTCAPTURE_3	Input Capture 3 select output
LL_RI_INPUTCAPTURE_4	Input Capture 4 select output

RI Input Capture Routing

LL_RI_INPUTCAPTUREROUTING_0	PA0 PA1 PA2 PA3
LL_RI_INPUTCAPTUREROUTING_1	PA4 PA5 PA6 PA7
LL_RI_INPUTCAPTUREROUTING_2	PA8 PA9 PA10 PA11
LL_RI_INPUTCAPTUREROUTING_3	PA12 PA13 PA14 PA15
LL_RI_INPUTCAPTUREROUTING_4	PC0 PC1 PC2 PC3
LL_RI_INPUTCAPTUREROUTING_5	PC4 PC5 PC6 PC7
LL_RI_INPUTCAPTUREROUTING_6	PC8 PC9 PC10 PC11
LL_RI_INPUTCAPTUREROUTING_7	PC12 PC13 PC14 PC15
LL_RI_INPUTCAPTUREROUTING_8	PD0 PD1 PD2 PD3
LL_RI_INPUTCAPTUREROUTING_9	PD4 PD5 PD6 PD7
LL_RI_INPUTCAPTUREROUTING_10	PD8 PD9 PD10 PD11
LL_RI_INPUTCAPTUREROUTING_11	PD12 PD13 PD14 PD15
LL_RI_INPUTCAPTUREROUTING_12	PE0 PE1 PE2 PE3
LL_RI_INPUTCAPTUREROUTING_13	PE4 PE5 PE6 PE7
LL_RI_INPUTCAPTUREROUTING_14	PE8 PE9 PE10 PE11
LL_RI_INPUTCAPTUREROUTING_15	PE12 PE13 PE14 PE15

RI IO Switch linked to ADC

LL_RI_IOSWITCH_CH0	CH[3:0] GR1[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH1	CH[3:0] GR1[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH2	CH[3:0] GR1[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH3	CH[3:0] GR1[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH4	CH4: Analog switch control
LL_RI_IOSWITCH_CH5	CH5: Comparator 1 analog switch
LL_RI_IOSWITCH_CH6	CH[7:6] GR2[2:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH7	CH[7:6] GR2[2:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH8	CH[9:8] GR3[2:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH9	CH[9:8] GR3[2:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH10	CH[13:10] GR8[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH11	CH[13:10] GR8[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH12	CH[13:10] GR8[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH13	CH[13:10] GR8[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH14	CH[15:14] GR9[2:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH15	CH[15:14] GR9[2:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH18	CH[21:18]/GR7[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH19	CH[21:18]/GR7[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH20	CH[21:18]/GR7[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH21	CH[21:18]/GR7[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH22	Analog I/O switch control of channels CH22
LL_RI_IOSWITCH_CH23	Analog I/O switch control of channels CH23
LL_RI_IOSWITCH_CH24	Analog I/O switch control of channels CH24
LL_RI_IOSWITCH_CH25	Analog I/O switch control of channels CH25
LL_RI_IOSWITCH_VCOMP	VCOMP (ADC channel 26) is an internal switch used to connect selected channel to COMP1 non inverting input
LL_RI_IOSWITCH_CH27	CH[30:27]/GR11[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH28	CH[30:27]/GR11[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH29	CH[30:27]/GR11[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH30	CH[30:27]/GR11[4:1]: I/O Analog switch control
LL_RI_IOSWITCH_CH31	CH31/GR11-5 I/O Analog switch control

RI IO Switch not linked to ADC

LL_RI_IOSWITCH_GR10_1	GR10-1 I/O analog switch control
LL_RI_IOSWITCH_GR10_2	GR10-2 I/O analog switch control
LL_RI_IOSWITCH_GR10_3	GR10-3 I/O analog switch control
LL_RI_IOSWITCH_GR10_4	GR10-4 I/O analog switch control
LL_RI_IOSWITCH_GR6_1	GR6-1 I/O analog switch control

LL_RI_IOSWITCH_GR6_2	GR6-2 I/O analog switch control
LL_RI_IOSWITCH_GR5_1	GR5-1 I/O analog switch control
LL_RI_IOSWITCH_GR5_2	GR5-2 I/O analog switch control
LL_RI_IOSWITCH_GR5_3	GR5-3 I/O analog switch control
LL_RI_IOSWITCH_GR4_1	GR4-1 I/O analog switch control
LL_RI_IOSWITCH_GR4_2	GR4-2 I/O analog switch control
LL_RI_IOSWITCH_GR4_3	GR4-3 I/O analog switch control
LL_RI_IOSWITCH_CH0b	CH0b-GR03-3 I/O analog switch control
LL_RI_IOSWITCH_CH1b	CH1b-GR03-4 I/O analog switch control
LL_RI_IOSWITCH_CH2b	CH2b-GR03-5 I/O analog switch control
LL_RI_IOSWITCH_CH3b	CH3b-GR09-3 I/O analog switch control
LL_RI_IOSWITCH_CH6b	CH6b-GR09-4 I/O analog switch control
LL_RI_IOSWITCH_CH7b	CH7b-GR02-3 I/O analog switch control
LL_RI_IOSWITCH_CH8b	CH8b-GR02-4 I/O analog switch control
LL_RI_IOSWITCH_CH9b	CH9b-GR02-5 I/O analog switch control
LL_RI_IOSWITCH_CH10b	CH10b-GR07-5 I/O analog switch control
LL_RI_IOSWITCH_CH11b	CH11b-GR07-6 I/O analog switch control
LL_RI_IOSWITCH_CH12b	CH12b-GR07-7 I/O analog switch control
LL_RI_IOSWITCH_GR6_3	GR6-3 I/O analog switch control
LL_RI_IOSWITCH_GR6_4	GR6-4 I/O analog switch control

FLASH LATENCY

LL_FLASH_LATENCY_0	FLASH Zero Latency cycle
LL_FLASH_LATENCY_1	FLASH One Latency cycle

SYSCFG LCD capacitance connection

LL_SYSCFG_LCDCAPA_PB2
LL_SYSCFG_LCDCAPA_PB12
LL_SYSCFG_LCDCAPA_PB0
LL_SYSCFG_LCDCAPA_PE11
LL_SYSCFG_LCDCAPA_PE12

RI PIN

LL_RI_PIN_0	Pin 0 selected
LL_RI_PIN_1	Pin 1 selected
LL_RI_PIN_2	Pin 2 selected
LL_RI_PIN_3	Pin 3 selected
LL_RI_PIN_4	Pin 4 selected
LL_RI_PIN_5	Pin 5 selected

LL_RI_PIN_6	Pin 6 selected
LL_RI_PIN_7	Pin 7 selected
LL_RI_PIN_8	Pin 8 selected
LL_RI_PIN_9	Pin 9 selected
LL_RI_PIN_10	Pin 10 selected
LL_RI_PIN_11	Pin 11 selected
LL_RI_PIN_12	Pin 12 selected
LL_RI_PIN_13	Pin 13 selected
LL_RI_PIN_14	Pin 14 selected
LL_RI_PIN_15	Pin 15 selected
LL_RI_PIN_ALL	All pins selected

RI PORT

LL_RI_PORT_A	PORt A
LL_RI_PORT_B	PORt B
LL_RI_PORT_C	PORt C
LL_RI_PORT_F	PORt F
LL_RI_PORT_G	PORt G

SYSCFG REMAP

LL_SYSCFG_REMAP_FLASH
LL_SYSCFG_REMAP_SYSTEMFLASH
LL_SYSCFG_REMAP_SRAM
LL_SYSCFG_REMAP_FMC

EXTI LINE

LL_SYSCFG_EXTI_LINE0
LL_SYSCFG_EXTI_LINE1
LL_SYSCFG_EXTI_LINE2
LL_SYSCFG_EXTI_LINE3
LL_SYSCFG_EXTI_LINE4
LL_SYSCFG_EXTI_LINE5
LL_SYSCFG_EXTI_LINE6
LL_SYSCFG_EXTI_LINE7
LL_SYSCFG_EXTI_LINE8
LL_SYSCFG_EXTI_LINE9
LL_SYSCFG_EXTI_LINE10
LL_SYSCFG_EXTI_LINE11
LL_SYSCFG_EXTI_LINE12

LL_SYSCFG_EXTI_LINE13
LL_SYSCFG_EXTI_LINE14
LL_SYSCFG_EXTI_LINE15

RI TIM selection

LL_RI_TIM_SELECT_NONE No timer selected
LL_RI_TIM_SELECT_TIM2 Timer 2 selected
LL_RI_TIM_SELECT_TIM3 Timer 3 selected
LL_RI_TIM_SELECT_TIM4 Timer 4 selected

DBGMCU TRACE Pin Assignment

LL_DBGMCU_TRACE_NONE	TRACE pins not assigned (default state)
LL_DBGMCU_TRACE_ASYNCH	TRACE pin assignment for Asynchronous Mode
LL_DBGMCU_TRACE_SYNCH_SIZE1	TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1
LL_DBGMCU_TRACE_SYNCH_SIZE2	TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2
LL_DBGMCU_TRACE_SYNCH_SIZE4	TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

66 LL TIM Generic Driver

66.1 TIM Firmware driver registers structures

66.1.1 LL_TIM_InitTypeDef

Data Fields

- *uint16_t Prescaler*
- *uint32_t CounterMode*
- *uint32_t Autoreload*
- *uint32_t ClockDivision*

Field Documentation

- ***uint16_t LL_TIM_InitTypeDef::Prescaler***
Specifies the prescaler value used to divide the TIM clock. This parameter can be a number between Min_Data=0x0000 and Max_Data=0xFFFF. This feature can be modified afterwards using unitary function **LL_TIM_SetPrescaler()**.
- ***uint32_t LL_TIM_InitTypeDef::CounterMode***
Specifies the counter mode. This parameter can be a value of **TIM_LL_EC_COUNTERMODE**. This feature can be modified afterwards using unitary function **LL_TIM_SetCounterMode()**.
- ***uint32_t LL_TIM_InitTypeDef::Autoreload***
Specifies the auto reload value to be loaded into the active Auto-Reload Register at the next update event. This parameter must be a number between Min_Data=0x0000 and Max_Data=0xFFFF. Some timer instances may support 32 bits counters. In that case this parameter must be a number between 0x0000 and 0xFFFFFFFF. This feature can be modified afterwards using unitary function **LL_TIM_SetAutoReload()**.
- ***uint32_t LL_TIM_InitTypeDef::ClockDivision***
Specifies the clock division. This parameter can be a value of **TIM_LL_EC_CLOCKDIVISION**. This feature can be modified afterwards using unitary function **LL_TIM_SetClockDivision()**.

66.1.2 LL_TIM_OC_InitTypeDef

Data Fields

- *uint32_t OCMode*
- *uint32_t OCState*
- *uint32_t CompareValue*
- *uint32_t OCPolarity*

Field Documentation

- ***uint32_t LL_TIM_OC_InitTypeDef::OCMode***
Specifies the output mode. This parameter can be a value of **TIM_LL_EC_OCMODE**. This feature can be modified afterwards using unitary function **LL_TIM_OC_SetMode()**.
- ***uint32_t LL_TIM_OC_InitTypeDef::OCState***
Specifies the TIM Output Compare state. This parameter can be a value of **TIM_LL_EC_OCSTATE**. This feature can be modified afterwards using unitary functions **LL_TIM_CC_EnableChannel()** or **LL_TIM_CC_DisableChannel()**.
- ***uint32_t LL_TIM_OC_InitTypeDef::CompareValue***
Specifies the Compare value to be loaded into the Capture Compare Register. This

- parameter can be a number between Min_Data=0x0000 and Max_Data=0xFFFF. This feature can be modified afterwards using unitary function LL_TIM_OC_SetCompareCHx (x=1..6).
- ***uint32_t LL_TIM_OC_InitTypeDef::OCPolarity***
Specifies the output polarity. This parameter can be a value of **TIM_LL_EC_OCPOLARITY**. This feature can be modified afterwards using unitary function **LL_TIM_OC_SetPolarity()**.

66.1.3 LL_TIM_IC_InitTypeDef

Data Fields

- ***uint32_t IC_Polarity***
- ***uint32_t IC_ActiveInput***
- ***uint32_t IC_Prescaler***
- ***uint32_t IC_Filter***

Field Documentation

- ***uint32_t LL_TIM_IC_InitTypeDef::IC_Polarity***
Specifies the active edge of the input signal. This parameter can be a value of **TIM_LL_EC_IC_POLARITY**. This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPolarity()**.
- ***uint32_t LL_TIM_IC_InitTypeDef::IC_ActiveInput***
Specifies the input. This parameter can be a value of **TIM_LL_EC_ACTIVEINPUT**. This feature can be modified afterwards using unitary function **LL_TIM_IC_SetActiveInput()**.
- ***uint32_t LL_TIM_IC_InitTypeDef::IC_Prescaler***
Specifies the Input Capture Prescaler. This parameter can be a value of **TIM_LL_EC_ICPSC**. This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPrescaler()**.
- ***uint32_t LL_TIM_IC_InitTypeDef::IC_Filter***
Specifies the input capture filter. This parameter can be a value of **TIM_LL_EC_IC_FILTER**. This feature can be modified afterwards using unitary function **LL_TIM_IC_SetFilter()**.

66.1.4 LL_TIM_ENCODER_InitTypeDef

Data Fields

- ***uint32_t EncoderMode***
- ***uint32_t IC1Polarity***
- ***uint32_t IC1ActiveInput***
- ***uint32_t IC1Prescaler***
- ***uint32_t IC1Filter***
- ***uint32_t IC2Polarity***
- ***uint32_t IC2ActiveInput***
- ***uint32_t IC2Prescaler***
- ***uint32_t IC2Filter***

Field Documentation

- ***uint32_t LL_TIM_ENCODER_InitTypeDef::EncoderMode***
Specifies the encoder resolution (x2 or x4). This parameter can be a value of **TIM_LL_EC_ENCODERMODE**. This feature can be modified afterwards using unitary function **LL_TIM_SetEncoderMode()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC1Polarity***
Specifies the active edge of TI1 input. This parameter can be a value of

- `TIM_LL_EC_IC_POLARITY`**. This feature can be modified afterwards using unitary function `LL_TIM_IC_SetPolarity()`.
- **`uint32_t LL_TIM_ENCODER_InitTypeDef::IC1ActiveInput`**
Specifies the TI1 input source. This parameter can be a value of **`TIM_LL_EC_ACTIVEINPUT`**. This feature can be modified afterwards using unitary function `LL_TIM_IC_SetActiveInput()`.
 - **`uint32_t LL_TIM_ENCODER_InitTypeDef::IC1Prescaler`**
Specifies the TI1 input prescaler value. This parameter can be a value of **`TIM_LL_EC_ICPSC`**. This feature can be modified afterwards using unitary function `LL_TIM_IC_SetPrescaler()`.
 - **`uint32_t LL_TIM_ENCODER_InitTypeDef::IC1Filter`**
Specifies the TI1 input filter. This parameter can be a value of **`TIM_LL_EC_IC_FILTER`**. This feature can be modified afterwards using unitary function `LL_TIM_IC_SetFilter()`.
 - **`uint32_t LL_TIM_ENCODER_InitTypeDef::IC2Polarity`**
Specifies the active edge of TI2 input. This parameter can be a value of **`TIM_LL_EC_IC_POLARITY`**. This feature can be modified afterwards using unitary function `LL_TIM_IC_SetPolarity()`.
 - **`uint32_t LL_TIM_ENCODER_InitTypeDef::IC2ActiveInput`**
Specifies the TI2 input source. This parameter can be a value of **`TIM_LL_EC_ACTIVEINPUT`**. This feature can be modified afterwards using unitary function `LL_TIM_IC_SetActiveInput()`.
 - **`uint32_t LL_TIM_ENCODER_InitTypeDef::IC2Prescaler`**
Specifies the TI2 input prescaler value. This parameter can be a value of **`TIM_LL_EC_ICPSC`**. This feature can be modified afterwards using unitary function `LL_TIM_IC_SetPrescaler()`.
 - **`uint32_t LL_TIM_ENCODER_InitTypeDef::IC2Filter`**
Specifies the TI2 input filter. This parameter can be a value of **`TIM_LL_EC_IC_FILTER`**. This feature can be modified afterwards using unitary function `LL_TIM_IC_SetFilter()`.

66.2 TIM Firmware driver API description

66.2.1 Detailed description of functions

LL_TIM_EnableCounter

Function name	<code>__STATIC_INLINE void LL_TIM_EnableCounter (TIM_TypeDef * TIMx)</code>
Function description	Enable timer counter.
Parameters	• TIMx: Timer instance
Return values	• None:
Reference Manual to LL API cross reference:	• CR1 CEN LL_TIM_EnableCounter

LL_TIM_DisableCounter

Function name	<code>__STATIC_INLINE void LL_TIM_DisableCounter (TIM_TypeDef * TIMx)</code>
Function description	Disable timer counter.

Parameters	<ul style="list-style-type: none"> TIMx: Timer instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 CEN LL_TIM_DisableCounter

LL_TIM_IsEnabledCounter

Function name	<code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledCounter((TIM_TypeDef * TIMx)</code>
Function description	Indicates whether the timer counter is enabled.
Parameters	<ul style="list-style-type: none"> TIMx: Timer instance
Return values	<ul style="list-style-type: none"> State: of bit (1 or 0).
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 CEN LL_TIM_IsEnabledCounter

LL_TIM_EnableUpdateEvent

Function name	<code>__STATIC_INLINE void LL_TIM_EnableUpdateEvent((TIM_TypeDef * TIMx)</code>
Function description	Enable update event generation.
Parameters	<ul style="list-style-type: none"> TIMx: Timer instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 UDIS LL_TIM_EnableUpdateEvent

LL_TIM_DisableUpdateEvent

Function name	<code>__STATIC_INLINE void LL_TIM_DisableUpdateEvent((TIM_TypeDef * TIMx)</code>
Function description	Disable update event generation.
Parameters	<ul style="list-style-type: none"> TIMx: Timer instance
Return values	<ul style="list-style-type: none"> None:
Reference Manual to LL API cross reference:	<ul style="list-style-type: none"> CR1 UDIS LL_TIM_DisableUpdateEvent

LL_TIM_IsEnabledUpdateEvent

Function name	<code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledUpdateEvent((TIM_TypeDef * TIMx)</code>
Function description	Indicates whether update event generation is enabled.
Parameters	<ul style="list-style-type: none"> TIMx: Timer instance

- Return values
- **State:** of bit (1 or 0).
- Reference Manual to
LL API cross
reference:
- CR1 UDIS LL_TIM_IsEnabledUpdateEvent

LL_TIM_SetUpdateSource

- Function name
- ```
_STATIC_INLINE void LL_TIM_SetUpdateSource
(TIM_TypeDef * TIMx, uint32_t UpdateSource)
```
- Function description
- Set update event source.
- Parameters
- **TIMx:** Timer instance
  - **UpdateSource:** This parameter can be one of the following values:
    - LL\_TIM\_UPDATESOURCE\_REGULAR
    - LL\_TIM\_UPDATESOURCE\_COUNTER
- Return values
- **None:**
- Notes
- Update event source set to LL\_TIM\_UPDATESOURCE\_REGULAR: any of the following events generate an update interrupt or DMA request if enabled: Counter overflow/underflowSetting the UG bitUpdate generation through the slave mode controller
  - Update event source set to LL\_TIM\_UPDATESOURCE\_COUNTER: only counter overflow/underflow generates an update interrupt or DMA request if enabled.
- Reference Manual to  
LL API cross  
reference:
- CR1 URS LL\_TIM\_SetUpdateSource

### **LL\_TIM\_GetUpdateSource**

- Function name
- ```
_STATIC_INLINE uint32_t LL_TIM_GetUpdateSource  
(TIM_TypeDef * TIMx)
```
- Function description
- Get actual event update source.
- Parameters
- **TIMx:** Timer instance
- Return values
- **Returned:** value can be one of the following values:
 - LL_TIM_UPDATESOURCE_REGULAR
 - LL_TIM_UPDATESOURCE_COUNTER
- Reference Manual to
LL API cross
reference:
- CR1 URS LL_TIM_GetUpdateSource

LL_TIM_SetOnePulseMode

- Function name
- ```
_STATIC_INLINE void LL_TIM_SetOnePulseMode
(TIM_TypeDef * TIMx, uint32_t OnePulseMode)
```
- Function description
- Set one pulse mode (one shot v.s.
- Parameters
- **TIMx:** Timer instance

- **OnePulseMode:** This parameter can be one of the following values:
    - LL\_TIM\_ONEPULSEMODE\_SINGLE
    - LL\_TIM\_ONEPULSEMODE\_REPETITIVE
  - **None:**
  - CR1 OPM LL\_TIM\_SetOnePulseMode
- Return values
- Reference Manual to LL API cross reference:

### **LL\_TIM\_SetOnePulseMode**

- |                                             |                                                                                                                                                                                       |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <b><code>_STATIC_INLINE uint32_t LL_TIM_SetOnePulseMode<br/>(TIM_TypeDef * TIMx)</code></b>                                                                                           |
| Function description                        | Get actual one pulse mode.                                                                                                                                                            |
| Parameters                                  | • <b>TIMx:</b> Timer instance                                                                                                                                                         |
| Return values                               | • <b>Returned:</b> value can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_ONEPULSEMODE_SINGLE</li> <li>- LL_TIM_ONEPULSEMODE_REPETITIVE</li> </ul> |
| Reference Manual to LL API cross reference: | • CR1 OPM LL_TIM_SetOnePulseMode                                                                                                                                                      |

### **LL\_TIM\_SetCounterMode**

- |                                             |                                                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <b><code>_STATIC_INLINE void LL_TIM_SetCounterMode<br/>(TIM_TypeDef * TIMx, uint32_t CounterMode)</code></b>                                                                                                                                                                                                                                          |
| Function description                        | Set the timer counter counting mode.                                                                                                                                                                                                                                                                                                                  |
| Parameters                                  | • <b>TIMx:</b> Timer instance<br>• <b>CounterMode:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_COUNTERMODE_UP</li> <li>- LL_TIM_COUNTERMODE_DOWN</li> <li>- LL_TIM_COUNTERMODE_CENTER_UP</li> <li>- LL_TIM_COUNTERMODE_CENTER_DOWN</li> <li>- LL_TIM_COUNTERMODE_CENTER_UP_DOWN</li> </ul> |
| Return values                               | • <b>None:</b>                                                                                                                                                                                                                                                                                                                                        |
| Notes                                       | • Macro IS_TIM_COUNTER_MODE_SELECT_INSTANCE(TIMx) can be used to check whether or not the counter mode selection feature is supported by a timer instance.                                                                                                                                                                                            |
| Reference Manual to LL API cross reference: | • CR1 DIR LL_TIM_SetCounterMode<br>• CR1 CMS LL_TIM_SetCounterMode                                                                                                                                                                                                                                                                                    |

### **LL\_TIM\_GetCounterMode**

- |               |                                                                                            |
|---------------|--------------------------------------------------------------------------------------------|
| Function name | <b><code>_STATIC_INLINE uint32_t LL_TIM_GetCounterMode<br/>(TIM_TypeDef * TIMx)</code></b> |
|---------------|--------------------------------------------------------------------------------------------|

|                                             |                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function description                        | Get actual counter mode.                                                                                                                                                                                                                                                                                                                                              |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                                                                                                                       |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_COUNTERMODE_UP</li> <li>- LL_TIM_COUNTERMODE_DOWN</li> <li>- LL_TIM_COUNTERMODE_CENTER_UP</li> <li>- LL_TIM_COUNTERMODE_CENTER_DOWN</li> <li>- LL_TIM_COUNTERMODE_CENTER_UP_DOWN</li> </ul> </li> </ul> |
| Notes                                       | <ul style="list-style-type: none"> <li>• Macro IS_TIM_COUNTER_MODE_SELECT_INSTANCE(TIMx) can be used to check whether or not the counter mode selection feature is supported by a timer instance.</li> </ul>                                                                                                                                                          |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CR1 DIR LL_TIM_GetCounterMode</li> <li>• CR1 CMS LL_TIM_GetCounterMode</li> </ul>                                                                                                                                                                                                                                            |

### LL\_TIM\_EnableARRPreload

|                                             |                                                                                      |
|---------------------------------------------|--------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE void LL_TIM_EnableARRPreload(<br/>  TIM_TypeDef * TIMx)</code> |
| Function description                        | Enable auto-reload (ARR) preload.                                                    |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>      |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                     |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CR1 ARPE LL_TIM_EnableARRPreload</li> </ul> |

### LL\_TIM\_DisableARRPreload

|                                             |                                                                                       |
|---------------------------------------------|---------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE void LL_TIM_DisableARRPreload(<br/>  TIM_TypeDef * TIMx)</code> |
| Function description                        | Disable auto-reload (ARR) preload.                                                    |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>       |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                      |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CR1 ARPE LL_TIM_DisableARRPreload</li> </ul> |

### LL\_TIM\_IsEnabledARRPreload

|                      |                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledARRPreload(<br/>  TIM_TypeDef * TIMx)</code> |
| Function description | Indicates whether auto-reload (ARR) preload is enabled.                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>             |
| Return values        | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>          |

- Reference Manual to LL API cross reference:
- CR1 ARPE LL\_TIM\_IsEnabledARRPreload

### **LL\_TIM\_SetClockDivision**

|                                             |                                                                                                                                                                                                                                                                                                                          |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <b><code>_STATIC_INLINE void LL_TIM_SetClockDivision (TIM_TypeDef * TIMx, uint32_t ClockDivision)</code></b>                                                                                                                                                                                                             |
| Function description                        | Set the division ratio between the timer clock and the sampling clock used by the dead-time generators (when supported) and the digital filters.                                                                                                                                                                         |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> <li><b>ClockDivision:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– LL_TIM_CLOCKDIVISION_DIV1</li> <li>– LL_TIM_CLOCKDIVISION_DIV2</li> <li>– LL_TIM_CLOCKDIVISION_DIV4</li> </ul> </li> </ul> |
| Return values                               | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                                                                                                                                                                           |
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_TIM_CLOCK_DIVISION_INSTANCE(TIMx) can be used to check whether or not the clock division feature is supported by the timer instance.</li> </ul>                                                                                                                          |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CR1 CKD LL_TIM_SetClockDivision</li> </ul>                                                                                                                                                                                                                                        |

### **LL\_TIM\_GetClockDivision**

|                                             |                                                                                                                                                                                                                                                                       |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <b><code>_STATIC_INLINE uint32_t LL_TIM_GetClockDivision (TIM_TypeDef * TIMx)</code></b>                                                                                                                                                                              |
| Function description                        | Get the actual division ratio between the timer clock and the sampling clock used by the dead-time generators (when supported) and the digital filters.                                                                                                               |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                         |
| Return values                               | <ul style="list-style-type: none"> <li><b>Returned:</b> value can be one of the following values: <ul style="list-style-type: none"> <li>– LL_TIM_CLOCKDIVISION_DIV1</li> <li>– LL_TIM_CLOCKDIVISION_DIV2</li> <li>– LL_TIM_CLOCKDIVISION_DIV4</li> </ul> </li> </ul> |
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_TIM_CLOCK_DIVISION_INSTANCE(TIMx) can be used to check whether or not the clock division feature is supported by the timer instance.</li> </ul>                                                                       |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CR1 CKD LL_TIM_GetClockDivision</li> </ul>                                                                                                                                                                                     |

### **LL\_TIM\_SetCounter**

|               |                                                                                                  |
|---------------|--------------------------------------------------------------------------------------------------|
| Function name | <b><code>_STATIC_INLINE void LL_TIM_SetCounter (TIM_TypeDef * TIMx, uint32_t Counter)</code></b> |
|---------------|--------------------------------------------------------------------------------------------------|

---

|                                                   |                                                                                                                                                                                 |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function description                              | Set the counter value.                                                                                                                                                          |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Counter:</b> Counter value (between Min_Data=0 and Max_Data=0xFFFF or 0xFFFFFFFF)</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> </ul>    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CNT CNT LL_TIM_SetCounter</li> </ul>                                                                                                   |

### LL\_TIM\_GetCounter

|                                                   |                                                                                                                                                                              |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_GetCounter (TIM_TypeDef * TIMx)</code>                                                                                                 |
| Function description                              | Get the counter value.                                                                                                                                                       |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                              |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Counter:</b> value (between Min_Data=0 and Max_Data=0xFFFF or 0xFFFFFFFF)</li> </ul>                                             |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CNT CNT LL_TIM_GetCounter</li> </ul>                                                                                                |

### LL\_TIM\_GetDirection

|                                                   |                                                                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_GetDirection (TIM_TypeDef * TIMx)</code>                                                                                                                                                                   |
| Function description                              | Get the current direction of the counter.                                                                                                                                                                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                  |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_COUNTERDIRECTION_UP</li> <li>- LL_TIM_COUNTERDIRECTION_DOWN</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 DIR LL_TIM_GetDirection</li> </ul>                                                                                                                                                                  |

### LL\_TIM\_SetPrescaler

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_SetPrescaler (TIM_TypeDef * TIMx, uint32_t Prescaler)</code> |
| Function description | Set the prescaler value.                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                |

- **Prescaler:** between Min\_Data=0 and Max\_Data=65535
- **None:**
- Notes
  - The counter clock frequency CK\_CNT is equal to fCK\_PSC / (PSC[15:0] + 1).
  - The prescaler can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.
  - Helper macro \_\_LL\_TIM\_CALC\_PSC can be used to calculate the Prescaler parameter
- Reference Manual to LL API cross reference:
  - PSC PSC LL\_TIM\_SetPrescaler

### **LL\_TIM\_GetPrescaler**

- Function name      **`__STATIC_INLINE uint32_t LL_TIM_GetPrescaler(  
                  TIM_TypeDef * TIMx)`**
- Function description      Get the prescaler value.
- Parameters      • **TIMx:** Timer instance
- Return values      • **Prescaler:** value between Min\_Data=0 and Max\_Data=65535
- Reference Manual to LL API cross reference:
  - PSC PSC LL\_TIM\_GetPrescaler

### **LL\_TIM\_SetAutoReload**

- Function name      **`__STATIC_INLINE void LL_TIM_SetAutoReload (TIM_TypeDef  
                  * TIMx, uint32_t AutoReload)`**
- Function description      Set the auto-reload value.
- Parameters      • **TIMx:** Timer instance  
• **AutoReload:** between Min\_Data=0 and Max\_Data=65535
- Return values      • **None:**
- Notes      • The counter is blocked while the auto-reload value is null.  
• Macro IS\_TIM\_32B\_COUNTER\_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.  
• Helper macro \_\_LL\_TIM\_CALC\_ARR can be used to calculate the AutoReload parameter
- Reference Manual to LL API cross reference:
  - ARR ARR LL\_TIM\_SetAutoReload

### **LL\_TIM\_GetAutoReload**

- Function name      **`__STATIC_INLINE uint32_t LL_TIM_GetAutoReload(  
                  TIM_TypeDef * TIMx)`**

---

|                                                   |                                                                                                                                                                              |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function description                              | Get the auto-reload value.                                                                                                                                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                              |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Auto-reload:</b> value</li> </ul>                                                                                                |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• ARR ARR LL_TIM_GetAutoReload</li> </ul>                                                                                             |

### LL\_TIM\_CC\_SetDMAReqTrigger

|                                                   |                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_CC_SetDMAReqTrigger(<br/>    TIM_TypeDef * TIMx, uint32_t DMAReqTrigger)</code>                                                                                                                                                                                 |
| Function description                              | Set the trigger of the capture/compare DMA request.                                                                                                                                                                                                                                               |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>DMAReqTrigger:</b> This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_TIM_CCDMAREQUEST_CC</li> <li>– LL_TIM_CCDMAREQUEST_UPDATE</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                  |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 CCDS LL_TIM_CC_SetDMAReqTrigger</li> </ul>                                                                                                                                                                                                           |

### LL\_TIM\_CC\_GetDMAReqTrigger

|                                                   |                                                                                                                                                                                                                                              |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_CC_GetDMAReqTrigger(<br/>    TIM_TypeDef * TIMx)</code>                                                                                                                                                |
| Function description                              | Get actual trigger of the capture/compare DMA request.                                                                                                                                                                                       |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                              |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_TIM_CCDMAREQUEST_CC</li> <li>– LL_TIM_CCDMAREQUEST_UPDATE</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 CCDS LL_TIM_CC_GetDMAReqTrigger</li> </ul>                                                                                                                                                      |

### LL\_TIM\_CC\_EnableChannel

|                      |                                                                                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_CC_EnableChannel(<br/>    TIM_TypeDef * TIMx, uint32_t Channels)</code>                                                                                                                                                    |
| Function description | Enable capture/compare channels.                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channels:</b> This parameter can be a combination of the following values:             <ul style="list-style-type: none"> <li>– LL_TIM_CHANNEL_CH1</li> </ul> </li> </ul> |

- LL\_TIM\_CHANNEL\_CH2
- LL\_TIM\_CHANNEL\_CH3
- LL\_TIM\_CHANNEL\_CH4

Return values

- **None:**

Reference Manual to  
LL API cross  
reference:

- CCER CC1E LL\_TIM\_CC\_EnableChannel
- CCER CC2E LL\_TIM\_CC\_EnableChannel
- CCER CC3E LL\_TIM\_CC\_EnableChannel
- CCER CC4E LL\_TIM\_CC\_EnableChannel

### **LL\_TIM\_CC\_DisableChannel**

Function name

**`_STATIC_INLINE void LL_TIM_CC_DisableChannel  
(TIM_TypeDef * TIMx, uint32_t Channels)`**

Function description

Disable capture/compare channels.

Parameters

- **TIMx:** Timer instance
- **Channels:** This parameter can be a combination of the following values:
  - LL\_TIM\_CHANNEL\_CH1
  - LL\_TIM\_CHANNEL\_CH2
  - LL\_TIM\_CHANNEL\_CH3
  - LL\_TIM\_CHANNEL\_CH4

Return values

- **None:**

Reference Manual to  
LL API cross  
reference:

- CCER CC1E LL\_TIM\_CC\_DisableChannel
- CCER CC2E LL\_TIM\_CC\_DisableChannel
- CCER CC3E LL\_TIM\_CC\_DisableChannel
- CCER CC4E LL\_TIM\_CC\_DisableChannel

### **LL\_TIM\_CC\_IsEnabledChannel**

Function name

**`_STATIC_INLINE uint32_t LL_TIM_CC_IsEnabledChannel  
(TIM_TypeDef * TIMx, uint32_t Channels)`**

Function description

Indicate whether channel(s) is(are) enabled.

Parameters

- **TIMx:** Timer instance
- **Channels:** This parameter can be a combination of the following values:
  - LL\_TIM\_CHANNEL\_CH1
  - LL\_TIM\_CHANNEL\_CH2
  - LL\_TIM\_CHANNEL\_CH3
  - LL\_TIM\_CHANNEL\_CH4

Return values

- **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:

- CCER CC1E LL\_TIM\_CC\_IsEnabledChannel
- CCER CC2E LL\_TIM\_CC\_IsEnabledChannel
- CCER CC3E LL\_TIM\_CC\_IsEnabledChannel
- CCER CC4E LL\_TIM\_CC\_IsEnabledChannel

**LL\_TIM\_OC\_ConfigOutput**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_OC_ConfigOutput<br/>(TIM_TypeDef * TIMx, uint32_t Channel, uint32_t<br/>Configuration)</code>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Function description                              | Configure an output channel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– LL_TIM_CHANNEL_CH1</li> <li>– LL_TIM_CHANNEL_CH2</li> <li>– LL_TIM_CHANNEL_CH3</li> <li>– LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>Configuration:</b> This parameter must be a combination of all the following values: <ul style="list-style-type: none"> <li>– LL_TIM_OCPOLARITY_HIGH or</li> <li>– LL_TIM_OCPOLARITY_LOW</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCMR1 CC1S LL_TIM_OC_ConfigOutput</li> <li>• CCMR1 CC2S LL_TIM_OC_ConfigOutput</li> <li>• CCMR2 CC3S LL_TIM_OC_ConfigOutput</li> <li>• CCMR2 CC4S LL_TIM_OC_ConfigOutput</li> <li>• CCER CC1P LL_TIM_OC_ConfigOutput</li> <li>• CCER CC2P LL_TIM_OC_ConfigOutput</li> <li>• CCER CC3P LL_TIM_OC_ConfigOutput</li> <li>• CCER CC4P LL_TIM_OC_ConfigOutput</li> <li>•</li> </ul>                                                                                                                            |

**LL\_TIM\_OC\_SetMode**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_OC_SetMode (TIM_TypeDef *<br/>TIMx, uint32_t Channel, uint32_t Mode)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Function description | Define the behavior of the output reference signal OCxREF from which OCx and OCxN (when relevant) are derived.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– LL_TIM_CHANNEL_CH1</li> <li>– LL_TIM_CHANNEL_CH2</li> <li>– LL_TIM_CHANNEL_CH3</li> <li>– LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>Mode:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– LL_TIM_OCMODE_FROZEN</li> <li>– LL_TIM_OCMODE_ACTIVE</li> <li>– LL_TIM_OCMODE_INACTIVE</li> <li>– LL_TIM_OCMODE_TOGGLE</li> <li>– LL_TIM_OCMODE_FORCED_INACTIVE</li> <li>– LL_TIM_OCMODE_FORCED_ACTIVE</li> <li>– LL_TIM_OCMODE_PWM1</li> <li>– LL_TIM_OCMODE_PWM2</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Reference Manual to  | <ul style="list-style-type: none"> <li>• CCMR1 OC1M LL_TIM_OC_SetMode</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

- LL API cross reference:
- CCMR1 OC2M LL\_TIM\_OC\_SetMode
  - CCMR2 OC3M LL\_TIM\_OC\_SetMode
  - CCMR2 OC4M LL\_TIM\_OC\_SetMode

### **LL\_TIM\_OC\_GetMode**

Function name **`__STATIC_INLINE uint32_t LL_TIM_OC_GetMode(  
 TIM_TypeDef * TIMx, uint32_t Channel)`**

Function description Get the output compare mode of an output channel.

- Parameters
- **TIMx:** Timer instance
  - **Channel:** This parameter can be one of the following values:
    - LL\_TIM\_CHANNEL\_CH1
    - LL\_TIM\_CHANNEL\_CH2
    - LL\_TIM\_CHANNEL\_CH3
    - LL\_TIM\_CHANNEL\_CH4
- Return values
- **Returned:** value can be one of the following values:
    - LL\_TIM\_OCMODE\_FROZEN
    - LL\_TIM\_OCMODE\_ACTIVE
    - LL\_TIM\_OCMODE\_INACTIVE
    - LL\_TIM\_OCMODE\_TOGGLE
    - LL\_TIM\_OCMODE\_FORCED\_INACTIVE
    - LL\_TIM\_OCMODE\_FORCED\_ACTIVE
    - LL\_TIM\_OCMODE\_PWM1
    - LL\_TIM\_OCMODE\_PWM2

- Reference Manual to  
LL API cross  
reference:
- CCMR1 OC1M LL\_TIM\_OC\_SetMode
  - CCMR1 OC2M LL\_TIM\_OC\_SetMode
  - CCMR2 OC3M LL\_TIM\_OC\_SetMode
  - CCMR2 OC4M LL\_TIM\_OC\_SetMode

### **LL\_TIM\_OC\_SetPolarity**

Function name **`__STATIC_INLINE void LL_TIM_OC_SetPolarity(  
 TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Polarity)`**

Function description Set the polarity of an output channel.

- Parameters
- **TIMx:** Timer instance
  - **Channel:** This parameter can be one of the following values:
    - LL\_TIM\_CHANNEL\_CH1
    - LL\_TIM\_CHANNEL\_CH2
    - LL\_TIM\_CHANNEL\_CH3
    - LL\_TIM\_CHANNEL\_CH4
  - **Polarity:** This parameter can be one of the following values:
    - LL\_TIM\_OCPOLARITY\_HIGH
    - LL\_TIM\_OCPOLARITY\_LOW

Return values

- Reference Manual to  
LL API cross  
reference:
- CCER CC1P LL\_TIM\_OC\_SetPolarity
  - CCER CC2P LL\_TIM\_OC\_SetPolarity
  - CCER CC3P LL\_TIM\_OC\_SetPolarity
  - CCER CC4P LL\_TIM\_OC\_SetPolarity

**LL\_TIM\_OC\_GetPolarity**

|                                                   |                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_OC_GetPolarity<br/>(TIM_TypeDef * TIMx, uint32_t Channel)</code>                                                                                                                                                                                                                          |
| Function description                              | Get the polarity of an output channel.                                                                                                                                                                                                                                                                                          |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_OCPOLARITY_HIGH</li> <li>- LL_TIM_OCPOLARITY_LOW</li> </ul> </li> </ul>                                                                                                     |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCER CC1P LL_TIM_OC_GetPolarity</li> <li>• CCER CC2P LL_TIM_OC_GetPolarity</li> <li>• CCER CC3P LL_TIM_OC_GetPolarity</li> <li>• CCER CC4P LL_TIM_OC_GetPolarity</li> </ul>                                                                                                            |

**LL\_TIM\_OC\_EnableFast**

|                                                   |                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_OC_EnableFast (TIM_TypeDef<br/>* TIMx, uint32_t Channel)</code>                                                                                                                                                                                                                               |
| Function description                              | Enable fast mode for the output channel.                                                                                                                                                                                                                                                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                |
| Notes                                             | <ul style="list-style-type: none"> <li>• Acts only if the channel is configured in PWM1 or PWM2 mode.</li> </ul>                                                                                                                                                                                                                |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCMR1 OC1FE LL_TIM_OC_EnableFast</li> <li>• CCMR1 OC2FE LL_TIM_OC_EnableFast</li> <li>• CCMR2 OC3FE LL_TIM_OC_EnableFast</li> <li>• CCMR2 OC4FE LL_TIM_OC_EnableFast</li> </ul>                                                                                                        |

**LL\_TIM\_OC\_DisableFast**

|                      |                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_OC_DisableFast<br/>(TIM_TypeDef * TIMx, uint32_t Channel)</code>                                                                                                                                                                                                |
| Function description | Disable fast mode for the output channel.                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> </ul> </li> </ul> |

– LL\_TIM\_CHANNEL\_CH4

**Return values**

- **None:**

**Reference Manual to  
LL API cross  
reference:**

- CCMR1 OC1FE LL\_TIM\_OC\_DisableFast
- CCMR1 OC2FE LL\_TIM\_OC\_DisableFast
- CCMR2 OC3FE LL\_TIM\_OC\_DisableFast
- CCMR2 OC4FE LL\_TIM\_OC\_DisableFast

### LL\_TIM\_OC\_IsEnabledFast

**Function name**

**`_STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledFast  
(TIM_TypeDef * TIMx, uint32_t Channel)`**

**Function description**

Indicates whether fast mode is enabled for the output channel.

**Parameters**

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL\_TIM\_CHANNEL\_CH1
  - LL\_TIM\_CHANNEL\_CH2
  - LL\_TIM\_CHANNEL\_CH3
  - LL\_TIM\_CHANNEL\_CH4

**Return values**

- **State:** of bit (1 or 0).

**Reference Manual to  
LL API cross  
reference:**

- CCMR1 OC1FE LL\_TIM\_OC\_IsEnabledFast
- CCMR1 OC2FE LL\_TIM\_OC\_IsEnabledFast
- CCMR2 OC3FE LL\_TIM\_OC\_IsEnabledFast
- CCMR2 OC4FE LL\_TIM\_OC\_IsEnabledFast
- 

### LL\_TIM\_OC\_EnablePreload

**Function name**

**`_STATIC_INLINE void LL_TIM_OC_EnablePreload  
(TIM_TypeDef * TIMx, uint32_t Channel)`**

**Function description**

Enable compare register (TIMx\_CCRx) preload for the output channel.

**Parameters**

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL\_TIM\_CHANNEL\_CH1
  - LL\_TIM\_CHANNEL\_CH2
  - LL\_TIM\_CHANNEL\_CH3
  - LL\_TIM\_CHANNEL\_CH4

**Return values**

- **None:**

**Reference Manual to  
LL API cross  
reference:**

- CCMR1 OC1PE LL\_TIM\_OC\_EnablePreload
- CCMR1 OC2PE LL\_TIM\_OC\_EnablePreload
- CCMR2 OC3PE LL\_TIM\_OC\_EnablePreload
- CCMR2 OC4PE LL\_TIM\_OC\_EnablePreload

### LL\_TIM\_OC\_DisablePreload

**Function name**

**`_STATIC_INLINE void LL_TIM_OC_DisablePreload  
(TIM_TypeDef * TIMx, uint32_t Channel)`**

|                                             |                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function description                        | Disable compare register (TIMx_CCRx) preload for the output channel.                                                                                                                                                                                                                                                            |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul> |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CCMR1 OC1PE LL_TIM_OC_DisablePreload</li> <li>• CCMR1 OC2PE LL_TIM_OC_DisablePreload</li> <li>• CCMR2 OC3PE LL_TIM_OC_DisablePreload</li> <li>• CCMR2 OC4PE LL_TIM_OC_DisablePreload</li> </ul>                                                                                        |

### LL\_TIM\_OC\_IsEnabledPreload

|                                             |                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <b>STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledPreload (TIM_TypeDef * TIMx, uint32_t Channel)</b>                                                                                                                                                                                                                                 |
| Function description                        | Indicates whether compare register (TIMx_CCRx) preload is enabled for the output channel.                                                                                                                                                                                                                                       |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul> |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                                                                                                                                                                              |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CCMR1 OC1PE LL_TIM_OC_IsEnabledPreload</li> <li>• CCMR1 OC2PE LL_TIM_OC_IsEnabledPreload</li> <li>• CCMR2 OC3PE LL_TIM_OC_IsEnabledPreload</li> <li>• CCMR2 OC4PE LL_TIM_OC_IsEnabledPreload</li> <li>•</li> </ul>                                                                     |

### LL\_TIM\_OC\_EnableClear

|                      |                                                                                                                                                                                                                                                                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>STATIC_INLINE void LL_TIM_OC_EnableClear (TIM_TypeDef * TIMx, uint32_t Channel)</b>                                                                                                                                                                                                                                          |
| Function description | Enable clearing the output channel on an external event.                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• This function can only be used in Output compare and PWM modes. It does not work in Forced mode.</li> <li>• Macro IS_TIM_OCXREF_CLEAR_INSTANCE(TIMx) can be</li> </ul>                                                                                                                 |

used to check whether or not a timer instance can clear the OCxREF signal on an external event.

Reference Manual to  
LL API cross  
reference:

- CCMR1 OC1CE LL\_TIM\_OC\_EnableClear
- CCMR1 OC2CE LL\_TIM\_OC\_EnableClear
- CCMR2 OC3CE LL\_TIM\_OC\_EnableClear
- CCMR2 OC4CE LL\_TIM\_OC\_EnableClear

### **LL\_TIM\_OC\_DisableClear**

|                                                   |                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_OC_DisableClear(<br/>(TIM_TypeDef * TIMx, uint32_t Channel)</code>                                                                                                                                                                                                                            |
| Function description                              | Disable clearing the output channel on an external event.                                                                                                                                                                                                                                                                       |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_OCXREF_CLEAR_INSTANCE(TIMx) can be used to check whether or not a timer instance can clear the OCxREF signal on an external event.</li> </ul>                                                                                                                             |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCMR1 OC1CE LL_TIM_OC_DisableClear</li> <li>• CCMR1 OC2CE LL_TIM_OC_DisableClear</li> <li>• CCMR2 OC3CE LL_TIM_OC_DisableClear</li> <li>• CCMR2 OC4CE LL_TIM_OC_DisableClear</li> </ul>                                                                                                |

### **LL\_TIM\_OC\_IsEnabledClear**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledClear(<br/>(TIM_TypeDef * TIMx, uint32_t Channel)</code>                                                                                                                                                                                                                                                                                         |
| Function description | Indicates clearing the output channel on an external event is enabled for the output channel.                                                                                                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul>                                                                    |
| Return values        | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                                                                                                                                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• This function enables clearing the output channel on an external event.</li> <li>• This function can only be used in Output compare and PWM modes. It does not work in Forced mode.</li> <li>• Macro IS_TIM_OCXREF_CLEAR_INSTANCE(TIMx) can be used to check whether or not a timer instance can clear the OCxREF signal on an external event.</li> </ul> |
| Reference Manual to  | <ul style="list-style-type: none"> <li>• CCMR1 OC1CE LL_TIM_OC_IsEnabledClear</li> </ul>                                                                                                                                                                                                                                                                                                           |

- LL API cross reference:
- CCMR1 OC2CE LL\_TIM\_OC\_IsEnabledClear
  - CCMR2 OC3CE LL\_TIM\_OC\_IsEnabledClear
  - CCMR2 OC4CE LL\_TIM\_OC\_IsEnabledClear
  -

### **LL\_TIM\_OC\_SetCompareCH1**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_OC_SetCompareCH1(<br/>TIM_TypeDef * TIMx, uint32_t CompareValue)</code></b>                                                                                                                                                                                                                                                                                                   |
| Function description                              | Set compare value for output channel 1 (TIMx_CCR1).                                                                                                                                                                                                                                                                                                                                                               |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>CompareValue:</b> between Min_Data=0 and Max_Data=65535</li> </ul>                                                                                                                                                                                                                                                             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                  |
| Notes                                             | <ul style="list-style-type: none"> <li>• In 32-bit timer implementations compare value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>• Macro IS_TIM_CC1_INSTANCE(TIMx) can be used to check whether or not output channel 1 is supported by a timer instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCR1 CCR1 LL_TIM_OC_SetCompareCH1</li> </ul>                                                                                                                                                                                                                                                                                                                             |

### **LL\_TIM\_OC\_SetCompareCH2**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_OC_SetCompareCH2(<br/>TIM_TypeDef * TIMx, uint32_t CompareValue)</code></b>                                                                                                                                                                                                                                                                                                   |
| Function description                              | Set compare value for output channel 2 (TIMx_CCR2).                                                                                                                                                                                                                                                                                                                                                               |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>CompareValue:</b> between Min_Data=0 and Max_Data=65535</li> </ul>                                                                                                                                                                                                                                                             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                  |
| Notes                                             | <ul style="list-style-type: none"> <li>• In 32-bit timer implementations compare value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>• Macro IS_TIM_CC2_INSTANCE(TIMx) can be used to check whether or not output channel 2 is supported by a timer instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCR2 CCR2 LL_TIM_OC_SetCompareCH2</li> </ul>                                                                                                                                                                                                                                                                                                                             |

**LL\_TIM\_OC\_SetCompareCH3**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_OC_SetCompareCH3<br/>(TIM_TypeDef * TIMx, uint32_t CompareValue)</code>                                                                                                                                                                                                                                                                                                       |
| Function description                              | Set compare value for output channel 3 (TIMx_CCR3).                                                                                                                                                                                                                                                                                                                                                             |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>CompareValue:</b> between Min_Data=0 and Max_Data=65535</li> </ul>                                                                                                                                                                                                                                                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                |
| Notes                                             | <ul style="list-style-type: none"> <li>• In 32-bit timer implementations compare value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>• Macro IS_TIM_CC3_INSTANCE(TIMx) can be used to check whether or not output channel is supported by a timer instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCR3 CCR3 LL_TIM_OC_SetCompareCH3</li> </ul>                                                                                                                                                                                                                                                                                                                           |

**LL\_TIM\_OC\_SetCompareCH4**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_OC_SetCompareCH4<br/>(TIM_TypeDef * TIMx, uint32_t CompareValue)</code>                                                                                                                                                                                                                                                                                                         |
| Function description                              | Set compare value for output channel 4 (TIMx_CCR4).                                                                                                                                                                                                                                                                                                                                                               |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>CompareValue:</b> between Min_Data=0 and Max_Data=65535</li> </ul>                                                                                                                                                                                                                                                             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                  |
| Notes                                             | <ul style="list-style-type: none"> <li>• In 32-bit timer implementations compare value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>• Macro IS_TIM_CC4_INSTANCE(TIMx) can be used to check whether or not output channel 4 is supported by a timer instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCR4 CCR4 LL_TIM_OC_SetCompareCH4</li> </ul>                                                                                                                                                                                                                                                                                                                             |

**LL\_TIM\_OC\_GetCompareCH1**

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH1<br/>(TIM_TypeDef * TIMx)</code> |
| Function description | Get compare value (TIMx_CCR1) set for output channel 1.                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>        |

|                                             |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Return values                               | <ul style="list-style-type: none"> <li>• <b>CompareValue:</b> (between Min_Data=0 and Max_Data=65535)</li> </ul>                                                                                                                                                                                                                                                                                                           |
| Notes                                       | <ul style="list-style-type: none"> <li>• In 32-bit timer implementations returned compare value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>• Macro IS_TIM_CC1_INSTANCE(TIMx) can be used to check whether or not output channel 1 is supported by a timer instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CCR1 CCR1 LL_TIM_OC_GetCompareCH1</li> </ul>                                                                                                                                                                                                                                                                                                                                      |

### LL\_TIM\_OC\_GetCompareCH2

|                                             |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH2(<br/>    TIM_TypeDef * TIMx)</code>                                                                                                                                                                                                                                                                                                                                 |
| Function description                        | Get compare value (TIMx_CCR2) set for output channel 2.                                                                                                                                                                                                                                                                                                                                                                    |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                                                                                                                                                                            |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>CompareValue:</b> (between Min_Data=0 and Max_Data=65535)</li> </ul>                                                                                                                                                                                                                                                                                                           |
| Notes                                       | <ul style="list-style-type: none"> <li>• In 32-bit timer implementations returned compare value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>• Macro IS_TIM_CC2_INSTANCE(TIMx) can be used to check whether or not output channel 2 is supported by a timer instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CCR2 CCR2 LL_TIM_OC_GetCompareCH2</li> </ul>                                                                                                                                                                                                                                                                                                                                      |

### LL\_TIM\_OC\_GetCompareCH3

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH3(<br/>    TIM_TypeDef * TIMx)</code>                                                                                                                                                                                                                                                                                                                       |
| Function description | Get compare value (TIMx_CCR3) set for output channel 3.                                                                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                                                                                                                                                                  |
| Return values        | <ul style="list-style-type: none"> <li>• <b>CompareValue:</b> (between Min_Data=0 and Max_Data=65535)</li> </ul>                                                                                                                                                                                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• In 32-bit timer implementations returned compare value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>• Macro IS_TIM_CC3_INSTANCE(TIMx) can be used to check whether or not output channel 3 is supported by a timer</li> </ul> |

instance.

Reference Manual to  
LL API cross  
reference:

- CCR3 CCR3 LL\_TIM\_OC\_GetCompareCH3

### **LL\_TIM\_OC\_GetCompareCH4**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH4(<br/>TIM_TypeDef * TIMx)</code></b>                                                                                                                                                                                                                                                                                                                               |
| Function description                              | Get compare value (TIMx_CCR4) set for output channel 4.                                                                                                                                                                                                                                                                                                                                                                    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                                                                                                                                                                            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>CompareValue:</b> (between Min_Data=0 and Max_Data=65535)</li> </ul>                                                                                                                                                                                                                                                                                                           |
| Notes                                             | <ul style="list-style-type: none"> <li>• In 32-bit timer implementations returned compare value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>• Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>• Macro IS_TIM_CC4_INSTANCE(TIMx) can be used to check whether or not output channel 4 is supported by a timer instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCR4 CCR4 LL_TIM_OC_GetCompareCH4</li> </ul>                                                                                                                                                                                                                                                                                                                                      |

### **LL\_TIM\_IC\_Config**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE void LL_TIM_IC_Config (TIM_TypeDef *<br/>TIMx, uint32_t Channel, uint32_t Configuration)</code></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Function description | Configure input channel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>Configuration:</b> This parameter must be a combination of all the following values: <ul style="list-style-type: none"> <li>- LL_TIM_ACTIVEINPUT_DIRECTTI or LL_TIM_ACTIVEINPUT_INDIRECTTI or LL_TIM_ACTIVEINPUT_TRC</li> <li>- LL_TIM_ICPSC_DIV1 or ... or LL_TIM_ICPSC_DIV8</li> <li>- LL_TIM_IC_FILTER_FDIV1 or ... or LL_TIM_IC_FILTER_FDIV32_N8</li> <li>- LL_TIM_IC_POLARITY_RISING or LL_TIM_IC_POLARITY_FALLING or LL_TIM_IC_POLARITY_BOTHEDGE</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Reference Manual to  | <ul style="list-style-type: none"> <li>• CCMR1 CC1S LL_TIM_IC_Config</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LL API cross reference: | <ul style="list-style-type: none"> <li>• CCMR1 IC1PSC LL_TIM_IC_Config</li> <li>• CCMR1 IC1F LL_TIM_IC_Config</li> <li>• CCMR1 CC2S LL_TIM_IC_Config</li> <li>• CCMR1 IC2PSC LL_TIM_IC_Config</li> <li>• CCMR1 IC2F LL_TIM_IC_Config</li> <li>• CCMR2 CC3S LL_TIM_IC_Config</li> <li>• CCMR2 IC3PSC LL_TIM_IC_Config</li> <li>• CCMR2 IC3F LL_TIM_IC_Config</li> <li>• CCMR2 CC4S LL_TIM_IC_Config</li> <li>• CCMR2 IC4PSC LL_TIM_IC_Config</li> <li>• CCMR2 IC4F LL_TIM_IC_Config</li> <li>• CCER CC1P LL_TIM_IC_Config</li> <li>• CCER CC1NP LL_TIM_IC_Config</li> <li>• CCER CC2P LL_TIM_IC_Config</li> <li>• CCER CC2NP LL_TIM_IC_Config</li> <li>• CCER CC3P LL_TIM_IC_Config</li> <li>• CCER CC3NP LL_TIM_IC_Config</li> <li>• CCER CC4P LL_TIM_IC_Config</li> <li>• CCER CC4NP LL_TIM_IC_Config</li> </ul> |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### LL\_TIM\_IC\_SetActiveInput

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_IC_SetActiveInput(<br/>    TIM_TypeDef * TIMx, uint32_t Channel, uint32_t<br/>    ICActiveInput)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Function description                              | Set the active input.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>ICActiveInput:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_ACTIVEINPUT_DIRECTTI</li> <li>- LL_TIM_ACTIVEINPUT_INDIRECTTI</li> <li>- LL_TIM_ACTIVEINPUT_TRC</li> </ul> </li> <li>• <b>None:</b></li> </ul> |
| Return values                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCMR1 CC1S LL_TIM_IC_SetActiveInput</li> <li>• CCMR1 CC2S LL_TIM_IC_SetActiveInput</li> <li>• CCMR2 CC3S LL_TIM_IC_SetActiveInput</li> <li>• CCMR2 CC4S LL_TIM_IC_SetActiveInput</li> </ul>                                                                                                                                                                                                                                                                                                                                                                    |

### LL\_TIM\_IC\_GetActiveInput

|                      |                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_IC_GetActiveInput(<br/>    TIM_TypeDef * TIMx, uint32_t Channel)</code>                                                 |
| Function description | Get the current active input.                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values:</li> </ul> |

|                                                   |                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                   | <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul>                                                                                                                     |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_ACTIVEINPUT_DIRECTTI</li> <li>- LL_TIM_ACTIVEINPUT_INDIRECTTI</li> <li>- LL_TIM_ACTIVEINPUT_TRC</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCMR1 CC1S LL_TIM_IC_GetActiveInput</li> <li>• CCMR1 CC2S LL_TIM_IC_GetActiveInput</li> <li>• CCMR2 CC3S LL_TIM_IC_GetActiveInput</li> <li>• CCMR2 CC4S LL_TIM_IC_GetActiveInput</li> </ul>                                                 |

### LL\_TIM\_IC\_SetPrescaler

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_IC_SetPrescaler(<br/>(TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICPrescaler)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Function description                              | Set the prescaler of input channel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>ICPrescaler:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_ICPSC_DIV1</li> <li>- LL_TIM_ICPSC_DIV2</li> <li>- LL_TIM_ICPSC_DIV4</li> <li>- LL_TIM_ICPSC_DIV8</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCMR1 IC1PSC LL_TIM_IC_SetPrescaler</li> <li>• CCMR1 IC2PSC LL_TIM_IC_SetPrescaler</li> <li>• CCMR2 IC3PSC LL_TIM_IC_SetPrescaler</li> <li>• CCMR2 IC4PSC LL_TIM_IC_SetPrescaler</li> </ul>                                                                                                                                                                                                                                                                                                                                                                |

### LL\_TIM\_IC\_GetPrescaler

|                      |                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_IC_GetPrescaler(<br/>(TIM_TypeDef * TIMx, uint32_t Channel)</code>                                                                                                                                                                                                                                  |
| Function description | Get the current prescaler value acting on an input channel.                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:</li> </ul>                                                                                                                                                                                                                            |

- LL\_TIM\_ICPSC\_DIV1
- LL\_TIM\_ICPSC\_DIV2
- LL\_TIM\_ICPSC\_DIV4
- LL\_TIM\_ICPSC\_DIV8

Reference Manual to  
LL API cross  
reference:

- CCMR1 IC1PSC LL\_TIM\_IC\_GetPrescaler
- CCMR1 IC2PSC LL\_TIM\_IC\_GetPrescaler
- CCMR2 IC3PSC LL\_TIM\_IC\_GetPrescaler
- CCMR2 IC4PSC LL\_TIM\_IC\_GetPrescaler

### **LL\_TIM\_IC\_SetFilter**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_IC_SetFilter (TIM_TypeDef *<br/>TIMx, uint32_t Channel, uint32_t ICFilter)</code></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Function description                              | Set the input filter duration.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>ICFilter:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_IC_FILTER_FDIV1</li> <li>- LL_TIM_IC_FILTER_FDIV1_N2</li> <li>- LL_TIM_IC_FILTER_FDIV1_N4</li> <li>- LL_TIM_IC_FILTER_FDIV1_N8</li> <li>- LL_TIM_IC_FILTER_FDIV2_N6</li> <li>- LL_TIM_IC_FILTER_FDIV2_N8</li> <li>- LL_TIM_IC_FILTER_FDIV4_N6</li> <li>- LL_TIM_IC_FILTER_FDIV4_N8</li> <li>- LL_TIM_IC_FILTER_FDIV8_N6</li> <li>- LL_TIM_IC_FILTER_FDIV8_N8</li> <li>- LL_TIM_IC_FILTER_FDIV16_N5</li> <li>- LL_TIM_IC_FILTER_FDIV16_N6</li> <li>- LL_TIM_IC_FILTER_FDIV16_N8</li> <li>- LL_TIM_IC_FILTER_FDIV32_N5</li> <li>- LL_TIM_IC_FILTER_FDIV32_N6</li> <li>- LL_TIM_IC_FILTER_FDIV32_N8</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCMR1 IC1F LL_TIM_IC_SetFilter</li> <li>• CCMR1 IC2F LL_TIM_IC_SetFilter</li> <li>• CCMR2 IC3F LL_TIM_IC_SetFilter</li> <li>• CCMR2 IC4F LL_TIM_IC_SetFilter</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

### **LL\_TIM\_IC\_GetFilter**

|                      |                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE uint32_t LL_TIM_IC_GetFilter (TIM_TypeDef<br/>* TIMx, uint32_t Channel)</code></b> |
| Function description | Get the input filter duration.                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                            |

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                   | <ul style="list-style-type: none"> <li>• <b>Channel:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_IC_FILTER_FDIV1</li> <li>- LL_TIM_IC_FILTER_FDIV1_N2</li> <li>- LL_TIM_IC_FILTER_FDIV1_N4</li> <li>- LL_TIM_IC_FILTER_FDIV1_N8</li> <li>- LL_TIM_IC_FILTER_FDIV2_N6</li> <li>- LL_TIM_IC_FILTER_FDIV2_N8</li> <li>- LL_TIM_IC_FILTER_FDIV4_N6</li> <li>- LL_TIM_IC_FILTER_FDIV4_N8</li> <li>- LL_TIM_IC_FILTER_FDIV8_N6</li> <li>- LL_TIM_IC_FILTER_FDIV8_N8</li> <li>- LL_TIM_IC_FILTER_FDIV16_N5</li> <li>- LL_TIM_IC_FILTER_FDIV16_N6</li> <li>- LL_TIM_IC_FILTER_FDIV16_N8</li> <li>- LL_TIM_IC_FILTER_FDIV32_N5</li> <li>- LL_TIM_IC_FILTER_FDIV32_N6</li> <li>- LL_TIM_IC_FILTER_FDIV32_N8</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCMR1 IC1F LL_TIM_IC_GetFilter</li> <li>• CCMR1 IC2F LL_TIM_IC_GetFilter</li> <li>• CCMR2 IC3F LL_TIM_IC_GetFilter</li> <li>• CCMR2 IC4F LL_TIM_IC_GetFilter</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

## LL\_TIM\_IC\_SetPolarity

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_IC_SetPolarity (TIM_TypeDef *<br/>TIMx, uint32_t Channel, uint32_t IC_Polarity)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Function description                              | Set the input channel polarity.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>IC_Polarity:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_TIM_IC_POLARITY_RISING</li> <li>- LL_TIM_IC_POLARITY_FALLING</li> <li>- LL_TIM_IC_POLARITY_BOTHEDGE</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCER CC1P LL_TIM_IC_SetPolarity</li> <li>• CCER CC1NP LL_TIM_IC_SetPolarity</li> <li>• CCER CC2P LL_TIM_IC_SetPolarity</li> <li>• CCER CC2NP LL_TIM_IC_SetPolarity</li> <li>• CCER CC3P LL_TIM_IC_SetPolarity</li> <li>• CCER CC3NP LL_TIM_IC_SetPolarity</li> </ul>                                                                                                                                                                                                                                                                                     |

- CCER CC4P LL\_TIM\_IC\_SetPolarity
- CCER CC4NP LL\_TIM\_IC\_SetPolarity

### LL\_TIM\_IC\_GetPolarity

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>_STATIC_INLINE uint32_t LL_TIM_IC_GetPolarity(<br/>    TIM_TypeDef * TIMx, uint32_t Channel)</code>                                                                                                                                                                                                                                                                                            |
| Function description                              | Get the current input channel polarity.                                                                                                                                                                                                                                                                                                                                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>- LL_TIM_CHANNEL_CH1</li> <li>- LL_TIM_CHANNEL_CH2</li> <li>- LL_TIM_CHANNEL_CH3</li> <li>- LL_TIM_CHANNEL_CH4</li> </ul> </li> </ul>                                                          |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:             <ul style="list-style-type: none"> <li>- LL_TIM_IC_POLARITY_RISING</li> <li>- LL_TIM_IC_POLARITY_FALLING</li> <li>- LL_TIM_IC_POLARITY_BOTHEDGE</li> </ul> </li> </ul>                                                                                                               |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CCER CC1P LL_TIM_IC_GetPolarity</li> <li>• CCER CC1NP LL_TIM_IC_GetPolarity</li> <li>• CCER CC2P LL_TIM_IC_GetPolarity</li> <li>• CCER CC2NP LL_TIM_IC_GetPolarity</li> <li>• CCER CC3P LL_TIM_IC_GetPolarity</li> <li>• CCER CC3NP LL_TIM_IC_GetPolarity</li> <li>• CCER CC4P LL_TIM_IC_GetPolarity</li> <li>• CCER CC4NP LL_TIM_IC_GetPolarity</li> </ul> |

### LL\_TIM\_IC\_EnableXORCombination

|                                                   |                                                                                                                                                                 |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>_STATIC_INLINE void LL_TIM_IC_EnableXORCombination(<br/>    TIM_TypeDef * TIMx)</code>                                                                    |
| Function description                              | Connect the TIMx_CH1, CH2 and CH3 pins to the TI1 input (XOR combination).                                                                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                 |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_XOR_INSTANCE(TIMx) can be used to check whether or not a timer instance provides an XOR input.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 TI1S LL_TIM_IC_EnableXORCombination</li> </ul>                                                                     |

### LL\_TIM\_IC\_DisableXORCombination

|                      |                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------|
| Function name        | <code>_STATIC_INLINE void LL_TIM_IC_DisableXORCombination(<br/>    TIM_TypeDef * TIMx)</code> |
| Function description | Disconnect the TIMx_CH1, CH2 and CH3 pins from the TI1 input.                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>               |

|                                                   |                                                                                                                                                               |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                |
| Notes                                             | <ul style="list-style-type: none"> <li>Macro IS_TIM_XOR_INSTANCE(TIMx) can be used to check whether or not a timer instance provides an XOR input.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR2 TI1S LL_TIM_IC_DisableXORCombination</li> </ul>                                                                    |

### LL\_TIM\_IC\_IsEnabledXORCombination

|                                                   |                                                                                                                                                               |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IC_IsEnabledXORCombination (TIM_TypeDef * TIMx)</code>                                                                  |
| Function description                              | Indicates whether the TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input.                                                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>                                                                                 |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>                                                                              |
| Notes                                             | <ul style="list-style-type: none"> <li>Macro IS_TIM_XOR_INSTANCE(TIMx) can be used to check whether or not a timer instance provides an XOR input.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR2 TI1S LL_TIM_IC_IsEnabledXORCombination</li> </ul>                                                                  |

### LL\_TIM\_IC\_GetCaptureCH1

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH1 (TIM_TypeDef * TIMx)</code>                                                                                                                                                                                                                                                                                                                                   |
| Function description                              | Get captured value for input channel 1.                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                                                                                                                                                                        |
| Return values                                     | <ul style="list-style-type: none"> <li><b>CapturedValue:</b> (between Min_Data=0 and Max_Data=65535)</li> </ul>                                                                                                                                                                                                                                                                                                      |
| Notes                                             | <ul style="list-style-type: none"> <li>In 32-bit timer implementations returned captured value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>Macro IS_TIM_CC1_INSTANCE(TIMx) can be used to check whether or not input channel 1 is supported by a timer instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CCR1 CCR1 LL_TIM_IC_GetCaptureCH1</li> </ul>                                                                                                                                                                                                                                                                                                                                  |

### LL\_TIM\_IC\_GetCaptureCH2

|                      |                                                                                    |
|----------------------|------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH2 (TIM_TypeDef * TIMx)</code> |
| Function description | Get captured value for input channel 2.                                            |
| Parameters           | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>      |

|                                             |                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Return values                               | <ul style="list-style-type: none"> <li><b>CapturedValue:</b> (between Min_Data=0 and Max_Data=65535)</li> </ul>                                                                                                                                                                                                                                                                                                      |
| Notes                                       | <ul style="list-style-type: none"> <li>In 32-bit timer implementations returned captured value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>Macro IS_TIM_CC2_INSTANCE(TIMx) can be used to check whether or not input channel 2 is supported by a timer instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CCR2 CCR2 LL_TIM_IC_GetCaptureCH2</li> </ul>                                                                                                                                                                                                                                                                                                                                  |

### LL\_TIM\_IC\_GetCaptureCH3

|                                             |                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH3(<br/>    TIM_TypeDef * TIMx)</code>                                                                                                                                                                                                                                                                                                                           |
| Function description                        | Get captured value for input channel 3.                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                                                                                                                                                                        |
| Return values                               | <ul style="list-style-type: none"> <li><b>CapturedValue:</b> (between Min_Data=0 and Max_Data=65535)</li> </ul>                                                                                                                                                                                                                                                                                                      |
| Notes                                       | <ul style="list-style-type: none"> <li>In 32-bit timer implementations returned captured value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>Macro IS_TIM_CC3_INSTANCE(TIMx) can be used to check whether or not input channel 3 is supported by a timer instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CCR3 CCR3 LL_TIM_IC_GetCaptureCH3</li> </ul>                                                                                                                                                                                                                                                                                                                                  |

### LL\_TIM\_IC\_GetCaptureCH4

|                      |                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH4(<br/>    TIM_TypeDef * TIMx)</code>                                                                                                                                                                                                                                                                                                                 |
| Function description | Get captured value for input channel 4.                                                                                                                                                                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                                                                                                                                                              |
| Return values        | <ul style="list-style-type: none"> <li><b>CapturedValue:</b> (between Min_Data=0 and Max_Data=65535)</li> </ul>                                                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>In 32-bit timer implementations returned captured value can be between 0x00000000 and 0xFFFFFFFF.</li> <li>Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.</li> <li>Macro IS_TIM_CC4_INSTANCE(TIMx) can be used to check whether or not input channel 4 is supported by a timer</li> </ul> |

instance.

Reference Manual to  
LL API cross  
reference:

- CCR4 CCR4 LL\_TIM\_IC\_GetCaptureCH4

### **LL\_TIM\_EnableExternalClock**

|                                                   |                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE void LL_TIM_EnableExternalClock(<br/>(TIM_TypeDef * TIMx)</code></b>                                                                                                                                                                                                                        |
| Function description                              | Enable external clock mode 2.                                                                                                                                                                                                                                                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                                                                                      |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                     |
| Notes                                             | <ul style="list-style-type: none"> <li>• When external clock mode 2 is enabled the counter is clocked by any active edge on the ETRF signal.</li> <li>• Macro <code>IS_TIM_CLOCKSOURCE_ETRMODE2_INSTANCE(TIMx)</code> can be used to check whether or not a timer instance supports external clock mode2.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SMCR ECE LL_TIM_EnableExternalClock</li> </ul>                                                                                                                                                                                                                              |

### **LL\_TIM\_DisableExternalClock**

|                                                   |                                                                                                                                                                                                       |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE void LL_TIM_DisableExternalClock(<br/>(TIM_TypeDef * TIMx)</code></b>                                                                                                        |
| Function description                              | Disable external clock mode 2.                                                                                                                                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                       |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                      |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro <code>IS_TIM_CLOCKSOURCE_ETRMODE2_INSTANCE(TIMx)</code> can be used to check whether or not a timer instance supports external clock mode2.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SMCR ECE LL_TIM_DisableExternalClock</li> </ul>                                                                                                              |

### **LL\_TIM\_IsEnabledExternalClock**

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledExternalClock(<br/>(TIM_TypeDef * TIMx)</code></b>                                                                   |
| Function description | Indicate whether external clock mode 2 is enabled.                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                        |
| Return values        | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• Macro <code>IS_TIM_CLOCKSOURCE_ETRMODE2_INSTANCE(TIMx)</code> can be used to check whether or not a timer instance</li> </ul> |

supports external clock mode2.

Reference Manual to  
LL API cross  
reference:

- SMCR ECE LL\_TIM\_IsEnabledExternalClock

### **LL\_TIM\_SetClockSource**

Function name

**`_STATIC_INLINE void LL_TIM_SetClockSource(  
(TIM_TypeDef * TIMx, uint32_t ClockSource)`**

Function description

Set the clock source of the counter clock.

Parameters

- **TIMx:** Timer instance
- **ClockSource:** This parameter can be one of the following values:
  - LL\_TIM\_CLOCKSOURCE\_INTERNAL
  - LL\_TIM\_CLOCKSOURCE\_EXT\_MODE1
  - LL\_TIM\_CLOCKSOURCE\_EXT\_MODE2

Return values

- **None:**

Notes

- when selected clock source is external clock mode 1, the timer input the external clock is applied is selected by calling the LL\_TIM\_SetTriggerInput() function. This timer input must be configured by calling the LL\_TIM\_IC\_Config() function.
- Macro IS\_TIM\_CLOCKSOURCE\_ETRMODE1\_INSTANCE(TIMx) can be used to check whether or not a timer instance supports external clock mode1.
- Macro IS\_TIM\_CLOCKSOURCE\_ETRMODE2\_INSTANCE(TIMx) can be used to check whether or not a timer instance supports external clock mode2.

Reference Manual to  
LL API cross  
reference:

- SMCR SMS LL\_TIM\_SetClockSource
- SMCR ECE LL\_TIM\_SetClockSource

### **LL\_TIM\_SetEncoderMode**

Function name

**`_STATIC_INLINE void LL_TIM_SetEncoderMode(  
(TIM_TypeDef * TIMx, uint32_t EncoderMode)`**

Function description

Set the encoder interface mode.

Parameters

- **TIMx:** Timer instance
- **EncoderMode:** This parameter can be one of the following values:
  - LL\_TIM\_ENCODERMODE\_X2\_TI1
  - LL\_TIM\_ENCODERMODE\_X2\_TI2
  - LL\_TIM\_ENCODERMODE\_X4\_TI12

Return values

- **None:**

Notes

- Macro IS\_TIM\_ENCODER\_INTERFACE\_INSTANCE(TIMx) can be used to check whether or not a timer instance supports the encoder mode.

- Reference Manual to  
LL API cross  
reference:
- SMCR SMS LL\_TIM\_SetEncoderMode

### **LL\_TIM\_SetTriggerOutput**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b>_STATIC_INLINE void LL_TIM_SetTriggerOutput<br/>(TIM_TypeDef * TIMx, uint32_t TimerSynchronization)</b>                                                                                                                                                                                                                                                                                                                                                         |
| Function description                              | Set the trigger output (TRGO) used for timer synchronization .                                                                                                                                                                                                                                                                                                                                                                                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>TimerSynchronization:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_TRGO_RESET</li> <li>- LL_TIM_TRGO_ENABLE</li> <li>- LL_TIM_TRGO_UPDATE</li> <li>- LL_TIM_TRGO_CC1IF</li> <li>- LL_TIM_TRGO_OC1REF</li> <li>- LL_TIM_TRGO_OC2REF</li> <li>- LL_TIM_TRGO_OC3REF</li> <li>- LL_TIM_TRGO_OC4REF</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                   |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_MASTER_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a master timer.</li> </ul>                                                                                                                                                                                                                                                                                         |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 MMS LL_TIM_SetTriggerOutput</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                |

### **LL\_TIM\_SetSlaveMode**

|                                                   |                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b>_STATIC_INLINE void LL_TIM_SetSlaveMode (TIM_TypeDef *<br/>TIMx, uint32_t SlaveMode)</b>                                                                                                                                                                                                                                                            |
| Function description                              | Set the synchronization mode of a slave timer.                                                                                                                                                                                                                                                                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>SlaveMode:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_SLAVEMODE_DISABLED</li> <li>- LL_TIM_SLAVEMODE_RESET</li> <li>- LL_TIM_SLAVEMODE_GATED</li> <li>- LL_TIM_SLAVEMODE_TRIGGER</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                       |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.</li> </ul>                                                                                                                                                                               |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SMCR SMS LL_TIM_SetSlaveMode</li> </ul>                                                                                                                                                                                                                                                                       |

**LL\_TIM\_SetTriggerInput**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_SetTriggerInput (TIM_TypeDef * TIMx, uint32_t TriggerInput)</code>                                                                                                                                                                                                                                                                                                                                |
| Function description                              | Set the selects the trigger input to be used to synchronize the counter.                                                                                                                                                                                                                                                                                                                                                            |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>TriggerInput:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_TS_ITR0</li> <li>- LL_TIM_TS_ITR1</li> <li>- LL_TIM_TS_ITR2</li> <li>- LL_TIM_TS_ITR3</li> <li>- LL_TIM_TS_TI1F_ED</li> <li>- LL_TIM_TS_TI1FP1</li> <li>- LL_TIM_TS_TI2FP2</li> <li>- LL_TIM_TS_ETRF</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.</li> </ul>                                                                                                                                                                                                                                                            |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SMCR TS LL_TIM_SetTriggerInput</li> </ul>                                                                                                                                                                                                                                                                                                                                                  |

**LL\_TIM\_EnableMasterSlaveMode**

|                                                   |                                                                                                                                                                          |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_EnableMasterSlaveMode (TIM_TypeDef * TIMx)</code>                                                                                      |
| Function description                              | Enable the Master/Slave mode.                                                                                                                                            |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                          |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                         |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SMCR MSM LL_TIM_EnableMasterSlaveMode</li> </ul>                                                                                |

**LL\_TIM\_DisableMasterSlaveMode**

|                      |                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_DisableMasterSlaveMode (TIM_TypeDef * TIMx)</code>                 |
| Function description | Disable the Master/Slave mode.                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                      |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                     |
| Notes                | <ul style="list-style-type: none"> <li>• Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to</li> </ul> |

check whether or not a timer instance can operate as a slave timer.

Reference Manual to  
LL API cross  
reference:

- SMCR MSM LL\_TIM\_DisableMasterSlaveMode

### **LL\_TIM\_IsEnabledMasterSlaveMode**

|                                                   |                                                                                                                                                                          |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledMasterSlaveMode (TIM_TypeDef * TIMx)</code></b>                                                                        |
| Function description                              | Indicates whether the Master/Slave mode is enabled.                                                                                                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                          |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                       |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SMCR MSM LL_TIM_IsEnabledMasterSlaveMode</li> </ul>                                                                             |

### **LL\_TIM\_ConfigETR**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE void LL_TIM_ConfigETR (TIM_TypeDef * TIMx, uint32_t ETRPolarity, uint32_t ETRPrescaler, uint32_t ETRFilter)</code></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Function description | Configure the external trigger (ETR) input.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>ETRPolarity:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_ETR_POLARITY_NONINVERTED</li> <li>- LL_TIM_ETR_POLARITY_INVERTED</li> </ul> </li> <li>• <b>ETRPrescaler:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_ETR_PRESCALER_DIV1</li> <li>- LL_TIM_ETR_PRESCALER_DIV2</li> <li>- LL_TIM_ETR_PRESCALER_DIV4</li> <li>- LL_TIM_ETR_PRESCALER_DIV8</li> </ul> </li> <li>• <b>ETRFilter:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_ETR_FILTER_FDIV1</li> <li>- LL_TIM_ETR_FILTER_FDIV1_N2</li> <li>- LL_TIM_ETR_FILTER_FDIV1_N4</li> <li>- LL_TIM_ETR_FILTER_FDIV1_N8</li> <li>- LL_TIM_ETR_FILTER_FDIV2_N6</li> <li>- LL_TIM_ETR_FILTER_FDIV2_N8</li> <li>- LL_TIM_ETR_FILTER_FDIV4_N6</li> <li>- LL_TIM_ETR_FILTER_FDIV4_N8</li> <li>- LL_TIM_ETR_FILTER_FDIV8_N6</li> <li>- LL_TIM_ETR_FILTER_FDIV8_N8</li> <li>- LL_TIM_ETR_FILTER_FDIV16_N5</li> </ul> </li> </ul> |

|                                                   |                                                                                                                                                                                                                                             |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                   | <ul style="list-style-type: none"> <li>- LL_TIM_ETR_FILTER_FDIV16_N6</li> <li>- LL_TIM_ETR_FILTER_FDIV16_N8</li> <li>- LL_TIM_ETR_FILTER_FDIV32_N5</li> <li>- LL_TIM_ETR_FILTER_FDIV32_N6</li> <li>- LL_TIM_ETR_FILTER_FDIV32_N8</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                            |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_TIM_ETR_INSTANCE(TIMx) can be used to check whether or not a timer instance provides an external trigger input.</li> </ul>                                                                |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SMCR ETP LL_TIM_ConfigETR</li> <li>• SMCR ETPS LL_TIM_ConfigETR</li> <li>• SMCR ETF LL_TIM_ConfigETR</li> </ul>                                                                                    |

### LL\_TIM\_ConfigDMAburst

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_ConfigDMAburst(<br/>    TIM_TypeDef * TIMx, uint32_t DMAburstBaseAddress,<br/>    uint32_t DMAburstLength)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Function description | Configures the timer DMA burst feature.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> <li>• <b>DMAburstBaseAddress:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_DMABURST_BASEADDR_CR1</li> <li>- LL_TIM_DMABURST_BASEADDR_CR2</li> <li>- LL_TIM_DMABURST_BASEADDR_SMCR</li> <li>- LL_TIM_DMABURST_BASEADDR_DIER</li> <li>- LL_TIM_DMABURST_BASEADDR_SR</li> <li>- LL_TIM_DMABURST_BASEADDR_EGR</li> <li>- LL_TIM_DMABURST_BASEADDR_CCMR1</li> <li>- LL_TIM_DMABURST_BASEADDR_CCMR2</li> <li>- LL_TIM_DMABURST_BASEADDR_CCER</li> <li>- LL_TIM_DMABURST_BASEADDR_CNT</li> <li>- LL_TIM_DMABURST_BASEADDR_PSC</li> <li>- LL_TIM_DMABURST_BASEADDR_ARR</li> <li>- LL_TIM_DMABURST_BASEADDR_CCR1</li> <li>- LL_TIM_DMABURST_BASEADDR_CCR2</li> <li>- LL_TIM_DMABURST_BASEADDR_CCR3</li> <li>- LL_TIM_DMABURST_BASEADDR_CCR4</li> <li>- LL_TIM_DMABURST_BASEADDR_OR</li> </ul> </li> <li>• <b>DMAburstLength:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_TIM_DMABURST_LENGTH_1TRANSFER</li> <li>- LL_TIM_DMABURST_LENGTH_2TRANSFERS</li> <li>- LL_TIM_DMABURST_LENGTH_3TRANSFERS</li> <li>- LL_TIM_DMABURST_LENGTH_4TRANSFERS</li> <li>- LL_TIM_DMABURST_LENGTH_5TRANSFERS</li> <li>- LL_TIM_DMABURST_LENGTH_6TRANSFERS</li> <li>- LL_TIM_DMABURST_LENGTH_7TRANSFERS</li> <li>- LL_TIM_DMABURST_LENGTH_8TRANSFERS</li> <li>- LL_TIM_DMABURST_LENGTH_9TRANSFERS</li> </ul> </li> </ul> |

- LL\_TIM\_DMABURST\_LENGTH\_10TRANSFERS
- LL\_TIM\_DMABURST\_LENGTH\_11TRANSFERS
- LL\_TIM\_DMABURST\_LENGTH\_12TRANSFERS
- LL\_TIM\_DMABURST\_LENGTH\_13TRANSFERS
- LL\_TIM\_DMABURST\_LENGTH\_14TRANSFERS
- LL\_TIM\_DMABURST\_LENGTH\_15TRANSFERS
- LL\_TIM\_DMABURST\_LENGTH\_16TRANSFERS
- LL\_TIM\_DMABURST\_LENGTH\_17TRANSFERS
- LL\_TIM\_DMABURST\_LENGTH\_18TRANSFERS

Return values

- **None:**

Notes

- Macro IS\_TIM\_DMABURST\_INSTANCE(TIMx) can be used to check whether or not a timer instance supports the DMA burst mode.

Reference Manual to  
LL API cross  
reference:

- DCR DBL LL\_TIM\_ConfigDMABurst
- DCR DBA LL\_TIM\_ConfigDMABurst

### **LL\_TIM\_SetRemap**

Function name

**`__STATIC_INLINE void LL_TIM_SetRemap (TIM_TypeDef *  
TIMx, uint32_t Remap)`**

Function description

Remap TIM inputs (input channel, internal/external triggers).

Parameters

- **TIMx:** Timer instance
- **Remap:** Remap params depends on the TIMx. Description available only in CHM version of the User Manual (not in .pdf). Otherwise see Reference Manual description of OR registers.

Return values

- **None:**

Notes

- Macro IS\_TIM\_REMAP\_INSTANCE(TIMx) can be used to check whether or not some timer inputs can be remapped.
- Option registers are available only for cat.3, cat.4 and cat.5 devices

Reference Manual to  
LL API cross  
reference:

- TIM2\_OR ITR1\_RMP LL\_TIM\_SetRemap
- TIM3\_OR ITR2\_RMP LL\_TIM\_SetRemap
- TIM9\_OR TI1\_RMP LL\_TIM\_SetRemap
- TIM9\_OR ITR1\_RMP LL\_TIM\_SetRemap
- TIM10\_OR TI1\_RMP LL\_TIM\_SetRemap
- TIM10\_OR ETR\_RMP LL\_TIM\_SetRemap
- TIM10\_OR TI1\_RMP\_RI LL\_TIM\_SetRemap
- TIM11\_OR TI1\_RMP LL\_TIM\_SetRemap
- TIM11\_OR ETR\_RMP LL\_TIM\_SetRemap
- TIM11\_OR TI1\_RMP\_RI LL\_TIM\_SetRemap

### **LL\_TIM\_SetOCRefClearInputSource**

Function name

**`__STATIC_INLINE void LL_TIM_SetOCRefClearInputSource  
(TIM_TypeDef * TIMx, uint32_t OCRefClearInputSource)`**

Function description

Set the OCREF clear input source.

|                                                   |                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> <li><b>OCRefClearInputSource:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_TIM_OCREF_CLR_INT_OCREF_CLR</li> <li>– LL_TIM_OCREF_CLR_INT_ETR</li> </ul> </li> </ul>                                 |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                                                                                                                                                                                            |
| Notes                                             | <ul style="list-style-type: none"> <li>The OCxREF signal of a given channel can be cleared when a high level is applied on the OCREF_CLR_INPUT</li> <li>This function can only be used in Output compare and PWM modes.</li> <li>the ETR signal can be connected to the output of a comparator to be used for current handling</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>SMCR OCCS LL_TIM_SetOCRefClearInputSource</li> </ul>                                                                                                                                                                                                                                               |

### LL\_TIM\_ClearFlag\_UPDATE

|                                                   |                                                                                   |
|---------------------------------------------------|-----------------------------------------------------------------------------------|
| Function name                                     | <code>_STATIC_INLINE void LL_TIM_ClearFlag_UPDATE<br/>(TIM_TypeDef * TIMx)</code> |
| Function description                              | Clear the update interrupt flag (UIF).                                            |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>     |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>SR UIF LL_TIM_ClearFlag_UPDATE</li> </ul>  |

### LL\_TIM\_IsActiveFlag\_UPDATE

|                                                   |                                                                                          |
|---------------------------------------------------|------------------------------------------------------------------------------------------|
| Function name                                     | <code>_STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_UPDATE<br/>(TIM_TypeDef * TIMx)</code> |
| Function description                              | Indicate whether update interrupt flag (UIF) is set (update interrupt is pending).       |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>            |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>         |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>SR UIF LL_TIM_IsActiveFlag_UPDATE</li> </ul>      |

### LL\_TIM\_ClearFlag\_CC1

|                      |                                                                                |
|----------------------|--------------------------------------------------------------------------------|
| Function name        | <code>_STATIC_INLINE void LL_TIM_ClearFlag_CC1 (TIM_TypeDef<br/>* TIMx)</code> |
| Function description | Clear the Capture/Compare 1 interrupt flag (CC1F).                             |
| Parameters           | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>  |
| Return values        | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                 |

- Reference Manual to  
LL API cross  
reference:
- SR CC1IF LL\_TIM\_ClearFlag\_CC1

### **LL\_TIM\_IsActiveFlag\_CC1**

|                                                   |                                                                                                           |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC1(<br/>    TIM_TypeDef * TIMx)</code>                |
| Function description                              | Indicate whether Capture/Compare 1 interrupt flag (CC1F) is set (Capture/Compare 1 interrupt is pending). |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                        |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CC1IF LL_TIM_IsActiveFlag_CC1</li> </ul>                      |

### **LL\_TIM\_ClearFlag\_CC2**

|                                                   |                                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_ClearFlag_CC2(<br/>    TIM_TypeDef * TIMx)</code> |
| Function description                              | Clear the Capture/Compare 2 interrupt flag (CC2F).                                  |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>     |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CC2IF LL_TIM_ClearFlag_CC2</li> </ul>   |

### **LL\_TIM\_IsActiveFlag\_CC2**

|                                                   |                                                                                                           |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC2(<br/>    TIM_TypeDef * TIMx)</code>                |
| Function description                              | Indicate whether Capture/Compare 2 interrupt flag (CC2F) is set (Capture/Compare 2 interrupt is pending). |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                        |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CC2IF LL_TIM_IsActiveFlag_CC2</li> </ul>                      |

### **LL\_TIM\_ClearFlag\_CC3**

|                      |                                                                                     |
|----------------------|-------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_ClearFlag_CC3(<br/>    TIM_TypeDef * TIMx)</code> |
| Function description | Clear the Capture/Compare 3 interrupt flag (CC3F).                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>     |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                    |

Reference Manual to  
LL API cross  
reference:

- SR CC3IF LL\_TIM\_ClearFlag\_CC3

### **LL\_TIM\_IsActiveFlag\_CC3**

Function name

**`_STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC3  
(TIM_TypeDef * TIMx)`**

Function description

Indicate whether Capture/Compare 3 interrupt flag (CC3F) is set (Capture/Compare 3 interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to

SR CC3IF LL\_TIM\_IsActiveFlag\_CC3

LL API cross

reference:

### **LL\_TIM\_ClearFlag\_CC4**

Function name

**`_STATIC_INLINE void LL_TIM_ClearFlag_CC4 (TIM_TypeDef  
* TIMx)`**

Function description

Clear the Capture/Compare 4 interrupt flag (CC4F).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to

SR CC4IF LL\_TIM\_ClearFlag\_CC4

LL API cross

reference:

### **LL\_TIM\_IsActiveFlag\_CC4**

Function name

**`_STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC4  
(TIM_TypeDef * TIMx)`**

Function description

Indicate whether Capture/Compare 4 interrupt flag (CC4F) is set (Capture/Compare 4 interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to

SR CC4IF LL\_TIM\_IsActiveFlag\_CC4

LL API cross

reference:

### **LL\_TIM\_ClearFlag\_TRIG**

Function name

**`_STATIC_INLINE void LL_TIM_ClearFlag_TRIG (TIM_TypeDef  
* TIMx)`**

Function description

Clear the trigger interrupt flag (TIF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

- Reference Manual to  
LL API cross  
reference:
- SR TIF LL\_TIM\_ClearFlag\_TRIG

### **LL\_TIM\_IsActiveFlag\_TRIG**

|                                                   |                                                                                         |
|---------------------------------------------------|-----------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_TRIG<br/>(TIM_TypeDef * TIMx)</code> |
| Function description                              | Indicate whether trigger interrupt flag (TIF) is set (trigger interrupt is pending).    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>         |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR TIF LL_TIM_IsActiveFlag_TRIG</li> </ul>     |

### **LL\_TIM\_ClearFlag\_CC1OVR**

|                                                   |                                                                                      |
|---------------------------------------------------|--------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_ClearFlag_CC1OVR<br/>(TIM_TypeDef * TIMx)</code>   |
| Function description                              | Clear the Capture/Compare 1 over-capture interrupt flag (CC1OF).                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>      |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                     |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CC1OF LL_TIM_ClearFlag_CC1OVR</li> </ul> |

### **LL\_TIM\_IsActiveFlag\_CC1OVR**

|                                                   |                                                                                                                         |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC1OVR<br/>(TIM_TypeDef * TIMx)</code>                               |
| Function description                              | Indicate whether Capture/Compare 1 over-capture interrupt flag (CC1OF) is set (Capture/Compare 1 interrupt is pending). |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                         |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CC1OF LL_TIM_IsActiveFlag_CC1OVR</li> </ul>                                 |

### **LL\_TIM\_ClearFlag\_CC2OVR**

|                      |                                                                                    |
|----------------------|------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_ClearFlag_CC2OVR<br/>(TIM_TypeDef * TIMx)</code> |
| Function description | Clear the Capture/Compare 2 over-capture interrupt flag (CC2OF).                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>    |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                   |

- Reference Manual to  
LL API cross  
reference:
- SR CC2OF LL\_TIM\_ClearFlag\_CC2OVR

### **LL\_TIM\_IsActiveFlag\_CC2OVR**

|                                                   |                                                                                                                                      |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC2OVR<br/>(TIM_TypeDef * TIMx)</code></b>                                      |
| Function description                              | Indicate whether Capture/Compare 2 over-capture interrupt flag (CC2OF) is set (Capture/Compare 2 over-capture interrupt is pending). |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                      |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CC2OF LL_TIM_IsActiveFlag_CC2OVR</li> </ul>                                              |

### **LL\_TIM\_ClearFlag\_CC3OVR**

|                                                   |                                                                                          |
|---------------------------------------------------|------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_ClearFlag_CC3OVR<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Clear the Capture/Compare 3 over-capture interrupt flag (CC3OF).                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>          |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                         |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CC3OF LL_TIM_ClearFlag_CC3OVR</li> </ul>     |

### **LL\_TIM\_IsActiveFlag\_CC3OVR**

|                                                   |                                                                                                                                      |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC3OVR<br/>(TIM_TypeDef * TIMx)</code></b>                                      |
| Function description                              | Indicate whether Capture/Compare 3 over-capture interrupt flag (CC3OF) is set (Capture/Compare 3 over-capture interrupt is pending). |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                      |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CC3OF LL_TIM_IsActiveFlag_CC3OVR</li> </ul>                                              |

### **LL\_TIM\_ClearFlag\_CC4OVR**

|                      |                                                                                          |
|----------------------|------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE void LL_TIM_ClearFlag_CC4OVR<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description | Clear the Capture/Compare 4 over-capture interrupt flag (CC4OF).                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>          |

---

|                                                   |                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------|
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                     |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>SR CC4OF LL_TIM_ClearFlag_CC4OVR</li> </ul> |

### **LL\_TIM\_IsActiveFlag\_CC4OVR**

|                                                   |                                                                                                                                      |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC4OVR<br/>(TIM_TypeDef * TIMx)</code></b>                                      |
| Function description                              | Indicate whether Capture/Compare 4 over-capture interrupt flag (CC4OF) is set (Capture/Compare 4 over-capture interrupt is pending). |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>                                                        |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>                                                     |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>SR CC4OF LL_TIM_IsActiveFlag_CC4OVR</li> </ul>                                                |

### **LL\_TIM\_EnableIT\_UPDATE**

|                                                   |                                                                                         |
|---------------------------------------------------|-----------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_EnableIT_UPDATE<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Enable update interrupt (UIE).                                                          |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>           |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER UIE LL_TIM_EnableIT_UPDATE</li> </ul>       |

### **LL\_TIM\_DisableIT\_UPDATE**

|                                                   |                                                                                          |
|---------------------------------------------------|------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_DisableIT_UPDATE<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Disable update interrupt (UIE).                                                          |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>            |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                           |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER UIE LL_TIM_DisableIT_UPDATE</li> </ul>       |

### **LL\_TIM\_IsEnabledIT\_UPDATE**

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_UPDATE<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description | Indicates whether the update interrupt (UIE) is enabled.                                       |
| Parameters           | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>                  |

---

|                                                   |                                                                                      |
|---------------------------------------------------|--------------------------------------------------------------------------------------|
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>     |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER UIE LL_TIM_IsEnabledIT_UPDATE</li> </ul> |

### LL\_TIM\_EnableIT\_CC1

|                                                   |                                                                                  |
|---------------------------------------------------|----------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_EnableIT_CC1 (TIM_TypeDef *<br/>TIMx)</code>   |
| Function description                              | Enable capture/compare 1 interrupt (CC1IE).                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>    |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC1IE LL_TIM_EnableIT_CC1</li> </ul> |

### LL\_TIM\_DisableIT\_CC1

|                                                   |                                                                                   |
|---------------------------------------------------|-----------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_DisableIT_CC1 (TIM_TypeDef *<br/>* TIMx)</code> |
| Function description                              | Disable capture/compare 1 interrupt (CC1IE).                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>     |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC1IE LL_TIM_DisableIT_CC1</li> </ul> |

### LL\_TIM\_IsEnabledIT\_CC1

|                                                   |                                                                                       |
|---------------------------------------------------|---------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC1<br/>(TIM_TypeDef * TIMx)</code> |
| Function description                              | Indicates whether the capture/compare 1 interrupt (CC1IE) is enabled.                 |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>         |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC1IE LL_TIM_IsEnabledIT_CC1</li> </ul>   |

### LL\_TIM\_EnableIT\_CC2

|                      |                                                                                |
|----------------------|--------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_EnableIT_CC2 (TIM_TypeDef *<br/>TIMx)</code> |
| Function description | Enable capture/compare 2 interrupt (CC2IE).                                    |
| Parameters           | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>  |

---

|                                                   |                                                                                  |
|---------------------------------------------------|----------------------------------------------------------------------------------|
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC2IE LL_TIM_EnableIT_CC2</li> </ul> |

### LL\_TIM\_DisableIT\_CC2

|                                                   |                                                                                   |
|---------------------------------------------------|-----------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_DisableIT_CC2 (TIM_TypeDef * TIMx)</code>       |
| Function description                              | Disable capture/compare 2 interrupt (CC2IE).                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>     |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC2IE LL_TIM_DisableIT_CC2</li> </ul> |

### LL\_TIM\_IsEnabledIT\_CC2

|                                                   |                                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC2 (TIM_TypeDef * TIMx)</code>   |
| Function description                              | Indicates whether the capture/compare 2 interrupt (CC2IE) is enabled.               |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>       |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC2IE LL_TIM_IsEnabledIT_CC2</li> </ul> |

### LL\_TIM\_EnableIT\_CC3

|                                                   |                                                                                  |
|---------------------------------------------------|----------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_EnableIT_CC3 (TIM_TypeDef * TIMx)</code>       |
| Function description                              | Enable capture/compare 3 interrupt (CC3IE).                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>    |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC3IE LL_TIM_EnableIT_CC3</li> </ul> |

### LL\_TIM\_DisableIT\_CC3

|                      |                                                                               |
|----------------------|-------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_DisableIT_CC3 (TIM_TypeDef * TIMx)</code>   |
| Function description | Disable capture/compare 3 interrupt (CC3IE).                                  |
| Parameters           | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul> |

---

|                                                   |                                                                                   |
|---------------------------------------------------|-----------------------------------------------------------------------------------|
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC3IE LL_TIM_DisableIT_CC3</li> </ul> |

### LL\_TIM\_IsEnabledIT\_CC3

|                                                   |                                                                                       |
|---------------------------------------------------|---------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC3<br/>(TIM_TypeDef * TIMx)</code> |
| Function description                              | Indicates whether the capture/compare 3 interrupt (CC3IE) is enabled.                 |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>         |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC3IE LL_TIM_IsEnabledIT_CC3</li> </ul>   |

### LL\_TIM\_EnableIT\_CC4

|                                                   |                                                                                  |
|---------------------------------------------------|----------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_EnableIT_CC4 (TIM_TypeDef *<br/>TIMx)</code>   |
| Function description                              | Enable capture/compare 4 interrupt (CC4IE).                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>    |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC4IE LL_TIM_EnableIT_CC4</li> </ul> |

### LL\_TIM\_DisableIT\_CC4

|                                                   |                                                                                   |
|---------------------------------------------------|-----------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_DisableIT_CC4 (TIM_TypeDef *<br/>TIMx)</code>   |
| Function description                              | Disable capture/compare 4 interrupt (CC4IE).                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>     |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC4IE LL_TIM_DisableIT_CC4</li> </ul> |

### LL\_TIM\_IsEnabledIT\_CC4

|                      |                                                                                       |
|----------------------|---------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC4<br/>(TIM_TypeDef * TIMx)</code> |
| Function description | Indicates whether the capture/compare 4 interrupt (CC4IE) is enabled.                 |
| Parameters           | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>         |

---

|                                                   |                                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------|
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER CC4IE LL_TIM_IsEnabledIT_CC4</li> </ul> |

### LL\_TIM\_EnableIT\_TRIG

|                                                   |                                                                                 |
|---------------------------------------------------|---------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_EnableIT_TRIG (TIM_TypeDef * TIMx)</code>     |
| Function description                              | Enable trigger interrupt (TIE).                                                 |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>   |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                  |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER TIE LL_TIM_EnableIT_TRIG</li> </ul> |

### LL\_TIM\_DisableIT\_TRIG

|                                                   |                                                                                  |
|---------------------------------------------------|----------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_TIM_DisableIT_TRIG (TIM_TypeDef * TIMx)</code>     |
| Function description                              | Disable trigger interrupt (TIE).                                                 |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>    |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER TIE LL_TIM_DisableIT_TRIG</li> </ul> |

### LL\_TIM\_IsEnabledIT\_TRIG

|                                                   |                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_TRIG (TIM_TypeDef * TIMx)</code> |
| Function description                              | Indicates whether the trigger interrupt (TIE) is enabled.                          |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>      |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DIER TIE LL_TIM_IsEnabledIT_TRIG</li> </ul> |

### LL\_TIM\_EnableDMAReq\_UPDATE

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_EnableDMAReq_UPDATE (TIM_TypeDef * TIMx)</code> |
| Function description | Enable update DMA request (UDE).                                                  |
| Parameters           | <ul style="list-style-type: none"> <li><b>TIMx:</b> Timer instance</li> </ul>     |
| Return values        | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                    |

- Reference Manual to  
LL API cross  
reference:
- DIER UDE LL\_TIM\_EnableDMAReq\_UPDATE

### **LL\_TIM\_DisableDMAReq\_UPDATE**

|                      |                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_DisableDMAReq_UPDATE(<br/>    TIM_TypeDef * TIMx)</code> |
| Function description | Disable update DMA request (UDE).                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>            |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                           |

Reference Manual to  
LL API cross  
reference:

- DIER UDE LL\_TIM\_DisableDMAReq\_UPDATE

### **LL\_TIM\_IsEnabledDMAReq\_UPDATE**

|                      |                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t<br/>LL_TIM_IsEnabledDMAReq_UPDATE(TIM_TypeDef * TIMx)</code> |
| Function description | Indicates whether the update DMA request (UDE) is enabled.                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>             |
| Return values        | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>          |

Reference Manual to  
LL API cross  
reference:

- DIER UDE LL\_TIM\_IsEnabledDMAReq\_UPDATE

### **LL\_TIM\_EnableDMAReq\_CC1**

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_EnableDMAReq_CC1(<br/>    TIM_TypeDef * TIMx)</code> |
| Function description | Enable capture/compare 1 DMA request (CC1DE).                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>        |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                       |

Reference Manual to  
LL API cross  
reference:

- DIER CC1DE LL\_TIM\_EnableDMAReq\_CC1

### **LL\_TIM\_DisableDMAReq\_CC1**

|                      |                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_TIM_DisableDMAReq_CC1(<br/>    TIM_TypeDef * TIMx)</code> |
| Function description | Disable capture/compare 1 DMA request (CC1DE).                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>         |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                        |

Reference Manual to  
LL API cross

- DIER CC1DE LL\_TIM\_DisableDMAReq\_CC1

reference:

### **LL\_TIM\_IsEnabledDMAReq\_CC1**

Function name      **STATIC\_INLINE uint32\_t LL\_TIM\_IsEnabledDMAReq\_CC1  
(TIM\_TypeDef \* TIMx)**

Function description      Indicates whether the capture/compare 1 DMA request (CC1DE) is enabled.

Parameters      • **TIMx:** Timer instance

Return values      • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:  
• DIER CC1DE LL\_TIM\_IsEnabledDMAReq\_CC1

### **LL\_TIM\_EnableDMAReq\_CC2**

Function name      **STATIC\_INLINE void LL\_TIM\_EnableDMAReq\_CC2  
(TIM\_TypeDef \* TIMx)**

Function description      Enable capture/compare 2 DMA request (CC2DE).

Parameters      • **TIMx:** Timer instance

Return values      • **None:**

Reference Manual to  
LL API cross  
reference:  
• DIER CC2DE LL\_TIM\_EnableDMAReq\_CC2

### **LL\_TIM\_DisableDMAReq\_CC2**

Function name      **STATIC\_INLINE void LL\_TIM\_DisableDMAReq\_CC2  
(TIM\_TypeDef \* TIMx)**

Function description      Disable capture/compare 2 DMA request (CC2DE).

Parameters      • **TIMx:** Timer instance

Return values      • **None:**

Reference Manual to  
LL API cross  
reference:  
• DIER CC2DE LL\_TIM\_DisableDMAReq\_CC2

### **LL\_TIM\_IsEnabledDMAReq\_CC2**

Function name      **STATIC\_INLINE uint32\_t LL\_TIM\_IsEnabledDMAReq\_CC2  
(TIM\_TypeDef \* TIMx)**

Function description      Indicates whether the capture/compare 2 DMA request (CC2DE) is enabled.

Parameters      • **TIMx:** Timer instance

Return values      • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:  
• DIER CC2DE LL\_TIM\_IsEnabledDMAReq\_CC2

reference:

### **LL\_TIM\_EnableDMAReq\_CC3**

|                                                   |                                                                                          |
|---------------------------------------------------|------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_EnableDMAReq_CC3<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Enable capture/compare 3 DMA request (CC3DE).                                            |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>          |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                         |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• DIER CC3DE LL_TIM_EnableDMAReq_CC3</li> </ul>   |

### **LL\_TIM\_DisableDMAReq\_CC3**

|                                                   |                                                                                           |
|---------------------------------------------------|-------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_DisableDMAReq_CC3<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Disable capture/compare 3 DMA request (CC3DE).                                            |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• DIER CC3DE LL_TIM_DisableDMAReq_CC3</li> </ul>   |

### **LL\_TIM\_IsEnabledDMAReq\_CC3**

|                                                   |                                                                                                 |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC3<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Indicates whether the capture/compare 3 DMA request (CC3DE) is enabled.                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                 |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>              |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• DIER CC3DE LL_TIM_IsEnabledDMAReq_CC3</li> </ul>       |

### **LL\_TIM\_EnableDMAReq\_CC4**

|                                     |                                                                                          |
|-------------------------------------|------------------------------------------------------------------------------------------|
| Function name                       | <b><code>_STATIC_INLINE void LL_TIM_EnableDMAReq_CC4<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                | Enable capture/compare 4 DMA request (CC4DE).                                            |
| Parameters                          | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>          |
| Return values                       | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                         |
| Reference Manual to<br>LL API cross | <ul style="list-style-type: none"> <li>• DIER CC4DE LL_TIM_EnableDMAReq_CC4</li> </ul>   |

reference:

### LL\_TIM\_DisableDMAReq\_CC4

Function name **`_STATIC_INLINE void LL_TIM_DisableDMAReq_CC4  
(TIM_TypeDef * TIMx)`**

Function description Disable capture/compare 4 DMA request (CC4DE).

Parameters • **TIMx:** Timer instance

Return values • **None:**

Reference Manual to  
LL API cross  
reference:  
DIER CC4DE LL\_TIM\_DisableDMAReq\_CC4

### LL\_TIM\_IsEnabledDMAReq\_CC4

Function name **`_STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC4  
(TIM_TypeDef * TIMx)`**

Function description Indicates whether the capture/compare 4 DMA request (CC4DE) is enabled.

Parameters • **TIMx:** Timer instance

Return values • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:  
DIER CC4DE LL\_TIM\_IsEnabledDMAReq\_CC4

### LL\_TIM\_EnableDMAReq\_TRIG

Function name **`_STATIC_INLINE void LL_TIM_EnableDMAReq_TRIG  
(TIM_TypeDef * TIMx)`**

Function description Enable trigger interrupt (TDE).

Parameters • **TIMx:** Timer instance

Return values • **None:**

Reference Manual to  
LL API cross  
reference:  
DIER TDE LL\_TIM\_EnableDMAReq\_TRIG

### LL\_TIM\_DisableDMAReq\_TRIG

Function name **`_STATIC_INLINE void LL_TIM_DisableDMAReq_TRIG  
(TIM_TypeDef * TIMx)`**

Function description Disable trigger interrupt (TDE).

Parameters • **TIMx:** Timer instance

Return values • **None:**

Reference Manual to  
LL API cross  
reference:  
DIER TDE LL\_TIM\_DisableDMAReq\_TRIG

reference:

### **LL\_TIM\_IsEnabledDMAReq\_TRIG**

|                                                   |                                                                                                           |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_TRIG(<br/>        TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Indicates whether the trigger interrupt (TDE) is enabled.                                                 |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                        |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• DIER TDE LL_TIM_IsEnabledDMAReq_TRIG</li> </ul>                  |

### **LL\_TIM\_GenerateEvent\_UPDATE**

|                                                   |                                                                                                       |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE void LL_TIM_GenerateEvent_UPDATE(<br/>        TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Generate an update event.                                                                             |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                       |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• EGR UG LL_TIM_GenerateEvent_UPDATE</li> </ul>                |

### **LL\_TIM\_GenerateEvent\_CC1**

|                                                   |                                                                                                    |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE void LL_TIM_GenerateEvent_CC1(<br/>        TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Generate Capture/Compare 1 event.                                                                  |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                    |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• EGR CC1G LL_TIM_GenerateEvent_CC1</li> </ul>              |

### **LL\_TIM\_GenerateEvent\_CC2**

|                                                   |                                                                                                    |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE void LL_TIM_GenerateEvent_CC2(<br/>        TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Generate Capture/Compare 2 event.                                                                  |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                    |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• EGR CC2G LL_TIM_GenerateEvent_CC2</li> </ul>              |

**LL\_TIM\_GenerateEvent\_CC3**

|                                                   |                                                                                           |
|---------------------------------------------------|-------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_GenerateEvent_CC3<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Generate Capture/Compare 3 event.                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• EGR CC3G LL_TIM_GenerateEvent_CC3</li> </ul>     |

**LL\_TIM\_GenerateEvent\_CC4**

|                                                   |                                                                                           |
|---------------------------------------------------|-------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_GenerateEvent_CC4<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Generate Capture/Compare 4 event.                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• EGR CC4G LL_TIM_GenerateEvent_CC4</li> </ul>     |

**LL\_TIM\_GenerateEvent\_TRIG**

|                                                   |                                                                                            |
|---------------------------------------------------|--------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_TIM_GenerateEvent_TRIG<br/>(TIM_TypeDef * TIMx)</code></b> |
| Function description                              | Generate trigger event.                                                                    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                           |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• EGR TG LL_TIM_GenerateEvent_TRIG</li> </ul>       |

**LL\_TIM\_DeInit**

|                      |                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>ErrorStatus LL_TIM_DeInit (TIM_TypeDef * TIMx)</code></b>                                                                                                                                                                              |
| Function description | Set TIMx registers to their reset values.                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer instance</li> </ul>                                                                                                                                                                 |
| Return values        | <ul style="list-style-type: none"> <li>• <b>An:</b> ErrorStatus enumeration value:           <ul style="list-style-type: none"> <li>– SUCCESS: TIMx registers are de-initialized</li> <li>– ERROR: invalid TIMx instance</li> </ul> </li> </ul> |

**LL\_TIM\_StructInit**

|                      |                                                                                  |
|----------------------|----------------------------------------------------------------------------------|
| Function name        | <b><code>void LL_TIM_StructInit (LL_TIM_InitTypeDef * TIM_InitStruct)</code></b> |
| Function description | Set the fields of the time base unit configuration data structure to             |

their default values.

- |               |                                                                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>TIM_InitStruct:</b> pointer to a LL_TIM_InitTypeDef structure (time base unit configuration data structure)</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                   |

### **LL\_TIM\_Init**

- |                      |                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>ErrorStatus LL_TIM_Init (TIM_TypeDef * TIMx,<br/>LL_TIM_InitTypeDef * TIM_InitStruct)</b>                                                                                                                                             |
| Function description | Configure the TIMx time base unit.                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer Instance</li> <li>• <b>TIM_InitStruct:</b> pointer to a LL_TIM_InitTypeDef structure (TIMx time base unit configuration data structure)</li> </ul>                           |
| Return values        | <ul style="list-style-type: none"> <li>• <b>An:</b> ErrorStatus enumeration value:           <ul style="list-style-type: none"> <li>– SUCCESS: TIMx registers are de-initialized</li> <li>– ERROR: not applicable</li> </ul> </li> </ul> |

### **LL\_TIM\_OC\_StructInit**

- |                      |                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>void LL_TIM_OC_StructInit (LL_TIM_OC_InitTypeDef *<br/>TIM_OC_InitStruct)</b>                                                                                             |
| Function description | Set the fields of the TIMx output channel configuration data structure to their default values.                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIM_OC_InitStruct:</b> pointer to a LL_TIM_OC_InitTypeDef structure (the output channel configuration data structure)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                             |

### **LL\_TIM\_OC\_Init**

- |                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>ErrorStatus LL_TIM_OC_Init (TIM_TypeDef * TIMx, uint32_t<br/>Channel, LL_TIM_OC_InitTypeDef * TIM_OC_InitStruct)</b>                                                                                                                                                                                                                                                                                                                                                        |
| Function description | Configure the TIMx output channel.                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer Instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_TIM_CHANNEL_CH1</li> <li>– LL_TIM_CHANNEL_CH2</li> <li>– LL_TIM_CHANNEL_CH3</li> <li>– LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>TIM_OC_InitStruct:</b> pointer to a LL_TIM_OC_InitTypeDef structure (TIMx output channel configuration data structure)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>An:</b> ErrorStatus enumeration value:           <ul style="list-style-type: none"> <li>– SUCCESS: TIMx output channel is initialized</li> <li>– ERROR: TIMx output channel is not initialized</li> </ul> </li> </ul>                                                                                                                                                                                                              |

### **LL\_TIM\_IC\_StructInit**

- |               |                                                           |
|---------------|-----------------------------------------------------------|
| Function name | <b>void LL_TIM_IC_StructInit (LL_TIM_IC_InitTypeDef *</b> |
|---------------|-----------------------------------------------------------|

**TIM\_ICInitStruct)**

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function description | Set the fields of the TIMx input channel configuration data structure to their default values.                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIM_ICInitStruct:</b> pointer to a LL_TIM_IC_InitTypeDef structure (the input channel configuration data structure)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                           |

**LL\_TIM\_IC\_Init**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>ErrorStatus LL_TIM_IC_Init (TIM_TypeDef * TIMx, uint32_t Channel, LL_TIM_IC_InitTypeDef * TIM_IC_InitStruct)</b>                                                                                                                                                                                                                                                                                                                                                 |
| Function description | Configure the TIMx input channel.                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer Instance</li> <li>• <b>Channel:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– LL_TIM_CHANNEL_CH1</li> <li>– LL_TIM_CHANNEL_CH2</li> <li>– LL_TIM_CHANNEL_CH3</li> <li>– LL_TIM_CHANNEL_CH4</li> </ul> </li> <li>• <b>TIM_IC_InitStruct:</b> pointer to a LL_TIM_IC_InitTypeDef structure (TIMx input channel configuration data structure)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>An:</b> ErrorStatus enumeration value: <ul style="list-style-type: none"> <li>– SUCCESS: TIMx output channel is initialized</li> <li>– ERROR: TIMx output channel is not initialized</li> </ul> </li> </ul>                                                                                                                                                                                                             |

**LL\_TIM\_ENCODER\_StructInit**

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>void LL_TIM_ENCODER_StructInit (LL_TIM_ENCODER_InitTypeDef * TIM_EncoderInitStruct)</b>                                                                                           |
| Function description | Fills each TIM_EncoderInitStruct field with its default value.                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIM_EncoderInitStruct:</b> pointer to a LL_TIM_ENCODER_InitTypeDef structure (encoder interface configuration data structure)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                     |

**LL\_TIM\_ENCODER\_Init**

|                      |                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>ErrorStatus LL_TIM_ENCODER_Init (TIM_TypeDef * TIMx, LL_TIM_ENCODER_InitTypeDef * TIM_EncoderInitStruct)</b>                                                                                                                  |
| Function description | Configure the encoder interface of the timer instance.                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TIMx:</b> Timer Instance</li> <li>• <b>TIM_EncoderInitStruct:</b> pointer to a LL_TIM_ENCODER_InitTypeDef structure (TIMx encoder interface configuration data structure)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>An:</b> ErrorStatus enumeration value: <ul style="list-style-type: none"> <li>– SUCCESS: TIMx registers are de-initialized</li> <li>– ERROR: not applicable</li> </ul> </li> </ul>   |

## 66.3 TIM Firmware driver defines

### 66.3.1 TIM

#### *Active Input Selection*

|                               |                      |
|-------------------------------|----------------------|
| LL_TIM_ACTIVEINPUT_DIRECTTI   | ICx is mapped on TIx |
| LL_TIM_ACTIVEINPUT_INDIRECTTI | ICx is mapped on Tly |
| LL_TIM_ACTIVEINPUT_TRC        | ICx is mapped on TRC |

#### *Capture Compare DMA Request*

|                            |                                                |
|----------------------------|------------------------------------------------|
| LL_TIM_CCDMAREQUEST_CC     | CCx DMA request sent when CCx event occurs     |
| LL_TIM_CCDMAREQUEST_UPDATE | CCx DMA requests sent when update event occurs |

#### *Channel*

|                    |                              |
|--------------------|------------------------------|
| LL_TIM_CHANNEL_CH1 | Timer input/output channel 1 |
| LL_TIM_CHANNEL_CH2 | Timer input/output channel 2 |
| LL_TIM_CHANNEL_CH3 | Timer input/output channel 3 |
| LL_TIM_CHANNEL_CH4 | Timer input/output channel 4 |

#### *Clock Division*

|                           |                |
|---------------------------|----------------|
| LL_TIM_CLOCKDIVISION_DIV1 | tDTS=tCK_INT   |
| LL_TIM_CLOCKDIVISION_DIV2 | tDTS=2*tCK_INT |
| LL_TIM_CLOCKDIVISION_DIV4 | tDTS=4*tCK_INT |

#### *Clock Source*

|                              |                                                                                 |
|------------------------------|---------------------------------------------------------------------------------|
| LL_TIM_CLOCKSOURCE_INTERNAL  | The timer is clocked by the internal clock provided from the RCC                |
| LL_TIM_CLOCKSOURCE_EXT_MODE1 | Counter counts at each rising or falling edge on a selected input               |
| LL_TIM_CLOCKSOURCE_EXT_MODE2 | Counter counts at each rising or falling edge on the external trigger input ETR |

#### *Counter Direction*

|                              |                           |
|------------------------------|---------------------------|
| LL_TIM_COUNTERDIRECTION_UP   | Timer counter counts up   |
| LL_TIM_COUNTERDIRECTION_DOWN | Timer counter counts down |

#### *Counter Mode*

|                                |                                                                                                                                                 |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| LL_TIM_COUNTERMODE_UP          | Counter used as upcounter                                                                                                                       |
| LL_TIM_COUNTERMODE_DOWN        | Counter used as downcounter                                                                                                                     |
| LL_TIM_COUNTERMODE_CENTER_UP   | The counter counts up and down alternatively. Output compare interrupt flags of output channels are set only when the counter is counting down. |
| LL_TIM_COUNTERMODE_CENTER_DOWN | The counter counts up and down alternatively. Output compare interrupt flags of output channels are set only                                    |

when the counter is counting up

|                                   |                                                                                                                                                       |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| LL_TIM_COUNTERMODE_CENTER_UP_DOWN | The counter counts up and down alternatively. Output compare interrupt flags of output channels are set only when the counter is counting up or down. |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

#### DMA Burst Base Address

|                                |                                                           |
|--------------------------------|-----------------------------------------------------------|
| LL_TIM_DMABURST_BASEADDR_CR1   | TIMx_CR1 register is the DMA base address for DMA burst   |
| LL_TIM_DMABURST_BASEADDR_CR2   | TIMx_CR2 register is the DMA base address for DMA burst   |
| LL_TIM_DMABURST_BASEADDR_SMCR  | TIMx_SMCR register is the DMA base address for DMA burst  |
| LL_TIM_DMABURST_BASEADDR_DIER  | TIMx_DIER register is the DMA base address for DMA burst  |
| LL_TIM_DMABURST_BASEADDR_SR    | TIMx_SR register is the DMA base address for DMA burst    |
| LL_TIM_DMABURST_BASEADDR_EGR   | TIMx_EGR register is the DMA base address for DMA burst   |
| LL_TIM_DMABURST_BASEADDR_CCMR1 | TIMx_CCMR1 register is the DMA base address for DMA burst |
| LL_TIM_DMABURST_BASEADDR_CCMR2 | TIMx_CCMR2 register is the DMA base address for DMA burst |
| LL_TIM_DMABURST_BASEADDR_CCER  | TIMx_CCER register is the DMA base address for DMA burst  |
| LL_TIM_DMABURST_BASEADDR_CNT   | TIMx_CNT register is the DMA base address for DMA burst   |
| LL_TIM_DMABURST_BASEADDR_PSC   | TIMx_PSC register is the DMA base address for DMA burst   |
| LL_TIM_DMABURST_BASEADDR_ARR   | TIMx_ARR register is the DMA base address for DMA burst   |
| LL_TIM_DMABURST_BASEADDR_CCR1  | TIMx_CCR1 register is the DMA base address for DMA burst  |
| LL_TIM_DMABURST_BASEADDR_CCR2  | TIMx_CCR2 register is the DMA base address for DMA burst  |
| LL_TIM_DMABURST_BASEADDR_CCR3  | TIMx_CCR3 register is the DMA base address for DMA burst  |
| LL_TIM_DMABURST_BASEADDR_CCR4  | TIMx_CCR4 register is the DMA base address for DMA burst  |
| LL_TIM_DMABURST_BASEADDR_OR    | TIMx_OR register is the DMA base address for DMA burst    |

#### DMA Burst Length

|                                  |                                                                         |
|----------------------------------|-------------------------------------------------------------------------|
| LL_TIM_DMABURST_LENGTH_1TRANSFER | Transfer is done to 1 register starting from the DMA burst base address |
|----------------------------------|-------------------------------------------------------------------------|

|                                    |                                                                           |
|------------------------------------|---------------------------------------------------------------------------|
| LL_TIM_DMABURST_LENGTH_2TRANSFERS  | Transfer is done to 2 registers starting from the DMA burst base address  |
| LL_TIM_DMABURST_LENGTH_3TRANSFERS  | Transfer is done to 3 registers starting from the DMA burst base address  |
| LL_TIM_DMABURST_LENGTH_4TRANSFERS  | Transfer is done to 4 registers starting from the DMA burst base address  |
| LL_TIM_DMABURST_LENGTH_5TRANSFERS  | Transfer is done to 5 registers starting from the DMA burst base address  |
| LL_TIM_DMABURST_LENGTH_6TRANSFERS  | Transfer is done to 6 registers starting from the DMA burst base address  |
| LL_TIM_DMABURST_LENGTH_7TRANSFERS  | Transfer is done to 7 registers starting from the DMA burst base address  |
| LL_TIM_DMABURST_LENGTH_8TRANSFERS  | Transfer is done to 1 registers starting from the DMA burst base address  |
| LL_TIM_DMABURST_LENGTH_9TRANSFERS  | Transfer is done to 9 registers starting from the DMA burst base address  |
| LL_TIM_DMABURST_LENGTH_10TRANSFERS | Transfer is done to 10 registers starting from the DMA burst base address |
| LL_TIM_DMABURST_LENGTH_11TRANSFERS | Transfer is done to 11 registers starting from the DMA burst base address |
| LL_TIM_DMABURST_LENGTH_12TRANSFERS | Transfer is done to 12 registers starting from the DMA burst base address |
| LL_TIM_DMABURST_LENGTH_13TRANSFERS | Transfer is done to 13 registers starting from the DMA burst base address |
| LL_TIM_DMABURST_LENGTH_14TRANSFERS | Transfer is done to 14 registers starting from the DMA burst base address |
| LL_TIM_DMABURST_LENGTH_15TRANSFERS | Transfer is done to 15 registers starting from the DMA burst base address |
| LL_TIM_DMABURST_LENGTH_16TRANSFERS | Transfer is done to 16 registers starting from the DMA burst base address |
| LL_TIM_DMABURST_LENGTH_17TRANSFERS | Transfer is done to 17 registers starting from the DMA burst base address |
| LL_TIM_DMABURST_LENGTH_18TRANSFERS | Transfer is done to 18 registers starting from the DMA burst base address |

**Encoder Mode**

|                            |                                                                                                                     |
|----------------------------|---------------------------------------------------------------------------------------------------------------------|
| LL_TIM_ENCODERMODE_X2_TI1  | Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level                                    |
| LL_TIM_ENCODERMODE_X2_TI2  | Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level                                    |
| LL_TIM_ENCODERMODE_X4_TI12 | Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input ! |

**External Trigger Filter**

|                         |                                     |
|-------------------------|-------------------------------------|
| LL_TIM_ETR_FILTER_FDIV1 | No filter, sampling is done at fDTS |
|-------------------------|-------------------------------------|

|                             |                        |
|-----------------------------|------------------------|
| LL_TIM_ETR_FILTER_FDIV1_N2  | fSAMPLING=fCK_INT, N=2 |
| LL_TIM_ETR_FILTER_FDIV1_N4  | fSAMPLING=fCK_INT, N=4 |
| LL_TIM_ETR_FILTER_FDIV1_N8  | fSAMPLING=fCK_INT, N=8 |
| LL_TIM_ETR_FILTER_FDIV2_N6  | fSAMPLING=fDTS/2, N=6  |
| LL_TIM_ETR_FILTER_FDIV2_N8  | fSAMPLING=fDTS/2, N=8  |
| LL_TIM_ETR_FILTER_FDIV4_N6  | fSAMPLING=fDTS/4, N=6  |
| LL_TIM_ETR_FILTER_FDIV4_N8  | fSAMPLING=fDTS/4, N=8  |
| LL_TIM_ETR_FILTER_FDIV8_N6  | fSAMPLING=fDTS/8, N=6  |
| LL_TIM_ETR_FILTER_FDIV8_N8  | fSAMPLING=fDTS/16, N=5 |
| LL_TIM_ETR_FILTER_FDIV16_N5 | fSAMPLING=fDTS/16, N=6 |
| LL_TIM_ETR_FILTER_FDIV16_N6 | fSAMPLING=fDTS/16, N=8 |
| LL_TIM_ETR_FILTER_FDIV16_N8 | fSAMPLING=fDTS/16, N=5 |
| LL_TIM_ETR_FILTER_FDIV32_N5 | fSAMPLING=fDTS/32, N=5 |
| LL_TIM_ETR_FILTER_FDIV32_N6 | fSAMPLING=fDTS/32, N=6 |
| LL_TIM_ETR_FILTER_FDIV32_N8 | fSAMPLING=fDTS/32, N=8 |

#### ***External Trigger Polarity***

|                                 |                                                          |
|---------------------------------|----------------------------------------------------------|
| LL_TIM_ETR_POLARITY_NONINVERTED | ETR is non-inverted, active at high level or rising edge |
| LL_TIM_ETR_POLARITY_INVERTED    | ETR is inverted, active at low level or falling edge     |

#### ***External Trigger Prescaler***

|                           |                               |
|---------------------------|-------------------------------|
| LL_TIM_ETR_PRESCALER_DIV1 | ETR prescaler OFF             |
| LL_TIM_ETR_PRESCALER_DIV2 | ETR frequency is divided by 2 |
| LL_TIM_ETR_PRESCALER_DIV4 | ETR frequency is divided by 4 |
| LL_TIM_ETR_PRESCALER_DIV8 | ETR frequency is divided by 8 |

#### ***Get Flags Defines***

|                 |                                    |
|-----------------|------------------------------------|
| LL_TIM_SR_UIF   | Update interrupt flag              |
| LL_TIM_SR_CC1IF | Capture/compare 1 interrupt flag   |
| LL_TIM_SR_CC2IF | Capture/compare 2 interrupt flag   |
| LL_TIM_SR_CC3IF | Capture/compare 3 interrupt flag   |
| LL_TIM_SR_CC4IF | Capture/compare 4 interrupt flag   |
| LL_TIM_SR_TIF   | Trigger interrupt flag             |
| LL_TIM_SR_CC1OF | Capture/Compare 1 overcapture flag |
| LL_TIM_SR_CC2OF | Capture/Compare 2 overcapture flag |
| LL_TIM_SR_CC3OF | Capture/Compare 3 overcapture flag |
| LL_TIM_SR_CC4OF | Capture/Compare 4 overcapture flag |

#### ***Input Configuration Prescaler***

**LL\_TIM\_ICPSC\_DIV1** No prescaler, capture is done each time an edge is detected on the capture input

**LL\_TIM\_ICPSC\_DIV2** Capture is done once every 2 events

**LL\_TIM\_ICPSC\_DIV4** Capture is done once every 4 events

**LL\_TIM\_ICPSC\_DIV8** Capture is done once every 8 events

#### ***Input Configuration Filter***

**LL\_TIM\_IC\_FILTER\_FDIV1** No filter, sampling is done at fDTS

**LL\_TIM\_IC\_FILTER\_FDIV1\_N2** fSAMPLING=fCK\_INT, N=2

**LL\_TIM\_IC\_FILTER\_FDIV1\_N4** fSAMPLING=fCK\_INT, N=4

**LL\_TIM\_IC\_FILTER\_FDIV1\_N8** fSAMPLING=fCK\_INT, N=8

**LL\_TIM\_IC\_FILTER\_FDIV2\_N6** fSAMPLING=fDTS/2, N=6

**LL\_TIM\_IC\_FILTER\_FDIV2\_N8** fSAMPLING=fDTS/2, N=8

**LL\_TIM\_IC\_FILTER\_FDIV4\_N6** fSAMPLING=fDTS/4, N=6

**LL\_TIM\_IC\_FILTER\_FDIV4\_N8** fSAMPLING=fDTS/4, N=8

**LL\_TIM\_IC\_FILTER\_FDIV8\_N6** fSAMPLING=fDTS/8, N=6

**LL\_TIM\_IC\_FILTER\_FDIV8\_N8** fSAMPLING=fDTS/8, N=8

**LL\_TIM\_IC\_FILTER\_FDIV16\_N5** fSAMPLING=fDTS/16, N=5

**LL\_TIM\_IC\_FILTER\_FDIV16\_N6** fSAMPLING=fDTS/16, N=6

**LL\_TIM\_IC\_FILTER\_FDIV16\_N8** fSAMPLING=fDTS/16, N=8

**LL\_TIM\_IC\_FILTER\_FDIV32\_N5** fSAMPLING=fDTS/32, N=5

**LL\_TIM\_IC\_FILTER\_FDIV32\_N6** fSAMPLING=fDTS/32, N=6

**LL\_TIM\_IC\_FILTER\_FDIV32\_N8** fSAMPLING=fDTS/32, N=8

#### ***Input Configuration Polarity***

**LL\_TIM\_IC\_POLARITY\_RISING** The circuit is sensitive to TIxFP1 rising edge, TIxFP1 is not inverted

**LL\_TIM\_IC\_POLARITY\_FALLING** The circuit is sensitive to TIxFP1 falling edge, TIxFP1 is inverted

**LL\_TIM\_IC\_POLARITY\_BOTHEDGE** The circuit is sensitive to both TIxFP1 rising and falling edges, TIxFP1 is not inverted

#### ***IT Defines***

**LL\_TIM\_DIER\_UIE** Update interrupt enable

**LL\_TIM\_DIER\_CC1IE** Capture/compare 1 interrupt enable

**LL\_TIM\_DIER\_CC2IE** Capture/compare 2 interrupt enable

**LL\_TIM\_DIER\_CC3IE** Capture/compare 3 interrupt enable

**LL\_TIM\_DIER\_CC4IE** Capture/compare 4 interrupt enable

**LL\_TIM\_DIER\_TIE** Trigger interrupt enable

#### ***Output Configuration Mode***

**LL\_TIM\_OCMODE\_FROZEN** The comparison between the output compare

|                               |                                                                                                                                                                            |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                               | register TIMx_CCRy and the counter<br>TIMx_CNT has no effect on the output channel<br>level                                                                                |
| LL_TIM_OCMODE_ACTIVE          | OCyREF is forced high on compare match                                                                                                                                     |
| LL_TIM_OCMODE_INACTIVE        | OCyREF is forced low on compare match                                                                                                                                      |
| LL_TIM_OCMODE_TOGGLE          | OCyREF toggles on compare match                                                                                                                                            |
| LL_TIM_OCMODE_FORCED_INACTIVE | OCyREF is forced low                                                                                                                                                       |
| LL_TIM_OCMODE_FORCED_ACTIVE   | OCyREF is forced high                                                                                                                                                      |
| LL_TIM_OCMODE_PWM1            | In upcounting, channel y is active as long as<br>TIMx_CNT<TIMx_CCRy else inactive. In<br>downcounting, channel y is inactive as long as<br>TIMx_CNT>TIMx_CCRy else active. |
| LL_TIM_OCMODE_PWM2            | In upcounting, channel y is inactive as long as<br>TIMx_CNT<TIMx_CCRy else active. In<br>downcounting, channel y is active as long as<br>TIMx_CNT>TIMx_CCRy else inactive  |

**Output Configuration Polarity**

|                        |                 |
|------------------------|-----------------|
| LL_TIM_OCPOLARITY_HIGH | OCx active high |
| LL_TIM_OCPOLARITY_LOW  | OCx active low  |

**OCREF clear input selection**

|                                |                                            |
|--------------------------------|--------------------------------------------|
| LL_TIM_OCREF_CLR_INT_OCREF_CLR | OCLR_INT is connected to the<br>OCLR input |
| LL_TIM_OCREF_CLR_INT_ETR       | OCLR_INT is connected to ETRF              |

**Output Configuration State**

|                        |                                                      |
|------------------------|------------------------------------------------------|
| LL_TIM_OCSTATE_DISABLE | OCx is not active                                    |
| LL_TIM_OCSTATE_ENABLE  | OCx signal is output on the corresponding output pin |

**One Pulse Mode**

|                                |                                                    |
|--------------------------------|----------------------------------------------------|
| LL_TIM_ONEPULSEMODE_SINGLE     | Counter is not stopped at update event             |
| LL_TIM_ONEPULSEMODE_REPETITIVE | Counter stops counting at the next update<br>event |

**Slave Mode**

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| LL_TIM_SLAVEMODE_DISABLED | Slave mode disabled                                                                        |
| LL_TIM_SLAVEMODE_RESET    | Reset Mode - Rising edge of the selected trigger<br>input (TRGI) reinitializes the counter |
| LL_TIM_SLAVEMODE_GATED    | Gated Mode - The counter clock is enabled when the<br>trigger input (TRGI) is high         |
| LL_TIM_SLAVEMODE_TRIGGER  | Trigger Mode - The counter starts at a rising edge of<br>the trigger TRGI                  |

**TIM10 ETR remap**

|                               |                                          |
|-------------------------------|------------------------------------------|
| LL_TIM_TIM10_ETR_RMP_LSE      | TIM10 ETR input is connected to LSE      |
| LL_TIM_TIM10_ETR_RMP_TIM9_TGO | TIM10 ETR input is connected to TIM9 TGO |

***TIM10 input 1 remapping capability***

|                           |                                                            |
|---------------------------|------------------------------------------------------------|
| LL_TIM_TIM10_TI1_RMP_GPIO | TIM10 channel1 is connected to GPIO                        |
| LL_TIM_TIM10_TI1_RMP_LSI  | TIM10 channel1 is connected to LSI internal clock          |
| LL_TIM_TIM10_TI1_RMP_LSE  | TIM10 channel1 is connected to LSE internal clock          |
| LL_TIM_TIM10_TI1_RMP_RTC  | TIM10 channel1 is connected to RTC wakeup interrupt signal |

***TIM10 Input 1 remap for Routing Interface (RI)***

|                         |                                                              |
|-------------------------|--------------------------------------------------------------|
| LL_TIM_TIM10_TI1_RMP    | TIM10 Channel1 connection depends on TI1_RMP[1:0] bit values |
| LL_TIM_TIM10_TI1_RMP_RI | TIM10 channel1 is connected to RI                            |

***TIM11 ETR remap***

|                               |                                                |
|-------------------------------|------------------------------------------------|
| LL_TIM_TIM11_ETR_RMP_LSE      | TIM11 ETR input is connected to LSE            |
| LL_TIM_TIM11_ETR_RMP_TIM9_TGO | TIM11 ETR input is connected to TIM9 TGO clock |

***TIM11 input 1 remapping capability***

|                              |                                                   |
|------------------------------|---------------------------------------------------|
| LL_TIM_TIM11_TI1_RMP_GPIO    | TIM11 channel1 is connected to GPIO               |
| LL_TIM_TIM11_TI1_RMP_MSI     | TIM11 channel1 is connected to MSI internal clock |
| LL_TIM_TIM11_TI1_RMP_HSE_RTC | TIM11 channel1 is connected to HSE RTC clock      |
| LL_TIM_TIM11_TI1_RMP_GPIO1   | TIM11 channel1 is connected to GPIO               |

***TIM11 Input 1 remap for Routing Interface (RI)***

|                         |                                                              |
|-------------------------|--------------------------------------------------------------|
| LL_TIM_TIM11_TI1_RMP    | TIM11 Channel1 connection depends on TI1_RMP[1:0] bit values |
| LL_TIM_TIM11_TI1_RMP_RI | TIM11 channel1 is connected to RI                            |

***TIM2 internal trigger 1 remap***

|                               |                                          |
|-------------------------------|------------------------------------------|
| LL_TIM_TIM2_TIR1_RMP_TIM10_OC | TIM2 ITR1 input is connected to TIM10 OC |
| LL_TIM_TIM2_TIR1_RMP_TIM5_TGO | TIM2 ITR1 input is connected to TIM5 TGO |

***TIM3 internal trigger 2 remap***

|                               |                                          |
|-------------------------------|------------------------------------------|
| LL_TIM_TIM3_TIR2_RMP_TIM11_OC | TIM3 ITR2 input is connected to TIM11 OC |
| LL_TIM_TIM3_TIR2_RMP_TIM5_TGO | TIM3 ITR2 input is connected to TIM5 TGO |

***TIM9 ITR1 remap***

|                               |                                                 |
|-------------------------------|-------------------------------------------------|
| LL_TIM_TIM9_ITR1_RMP_TIM3_TGO | TIM9 channel1 is connected to TIM3 TGO signal   |
| LL_TIM_TIM9_ITR1_RMP_TOUCH_IO | TIM9 channel1 is connected to touch sensing I/O |

***TIM9 Input 1 remap***

|                           |                                                  |
|---------------------------|--------------------------------------------------|
| LL_TIM_TIM9_TI1_RMP_GPIO  | TIM9 channel1 is connected to GPIO               |
| LL_TIM_TIM9_TI1_RMP_LSE   | TIM9 channel1 is connected to LSE internal clock |
| LL_TIM_TIM9_TI1_RMP_GPIO1 | TIM9 channel1 is connected to GPIO               |
| LL_TIM_TIM9_TI1_RMP_GPIO2 | TIM9 channel1 is connected to GPIO               |

***Trigger Output***

|                                 |                                                             |
|---------------------------------|-------------------------------------------------------------|
| <code>LL_TIM_TRGO_RESET</code>  | UG bit from the TIMx_EGR register is used as trigger output |
| <code>LL_TIM_TRGO_ENABLE</code> | Counter Enable signal (CNT_EN) is used as trigger output    |
| <code>LL_TIM_TRGO_UPDATE</code> | Update event is used as trigger output                      |
| <code>LL_TIM_TRGO_CC1IF</code>  | CC1 capture or a compare match is used as trigger output    |
| <code>LL_TIM_TRGO_OC1REF</code> | OC1REF signal is used as trigger output                     |
| <code>LL_TIM_TRGO_OC2REF</code> | OC2REF signal is used as trigger output                     |
| <code>LL_TIM_TRGO_OC3REF</code> | OC3REF signal is used as trigger output                     |
| <code>LL_TIM_TRGO_OC4REF</code> | OC4REF signal is used as trigger output                     |

***Trigger Selection***

|                                |                                                           |
|--------------------------------|-----------------------------------------------------------|
| <code>LL_TIM_TS_ITR0</code>    | Internal Trigger 0 (ITR0) is used as trigger input        |
| <code>LL_TIM_TS_ITR1</code>    | Internal Trigger 1 (ITR1) is used as trigger input        |
| <code>LL_TIM_TS_ITR2</code>    | Internal Trigger 2 (ITR2) is used as trigger input        |
| <code>LL_TIM_TS_ITR3</code>    | Internal Trigger 3 (ITR3) is used as trigger input        |
| <code>LL_TIM_TS_TI1F_ED</code> | TI1 Edge Detector (TI1F_ED) is used as trigger input      |
| <code>LL_TIM_TS_TI1FP1</code>  | Filtered Timer Input 1 (TI1FP1) is used as trigger input  |
| <code>LL_TIM_TS_TI2FP2</code>  | Filtered Timer Input 2 (TI12P2) is used as trigger input  |
| <code>LL_TIM_TS_ETRF</code>    | Filtered external Trigger (ETRF) is used as trigger input |

***Update Source***

|                                          |                                                                                                                                   |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <code>LL_TIM_UPDATESOURCE_REGULAR</code> | Counter overflow/underflow, Setting the UG bit or Update generation through the slave mode controller generates an update request |
| <code>LL_TIM_UPDATESOURCE_COUNTER</code> | Only counter overflow/underflow generates an update request                                                                       |

***Exported Macros***

- `__LL_TIM_CALC_PSC`      **Description:**
- HELPER macro calculating the prescaler value to achieve the required counter clock frequency.

**Parameters:**

- `__TIMCLK__`: timer input clock frequency (in Hz)
- `__CNTCLK__`: counter clock frequency (in Hz)

**Return value:**

- Prescaler: value (between Min\_Data=0 and Max\_Data=65535)

**Notes:**

- ex: `__LL_TIM_CALC_PSC (80000000, 1000000)`;

`__LL_TIM_CALC_ARR`**Description:**

- HELPER macro calculating the auto-reload value to achieve the required output signal frequency.

**Parameters:**

- `__TIMCLK__`: timer input clock frequency (in Hz)
- `__PSC__`: prescaler
- `__FREQ__`: output signal frequency (in Hz)

**Return value:**

- Auto-reload: value (between Min\_Data=0 and Max\_Data=65535)

**Notes:**

- ex: `__LL_TIM_CALC_ARR (1000000, LL_TIM_GetPrescaler (), 10000);`

**\_\_LL\_TIM\_CALC\_DELAY****Description:**

- HELPER macro calculating the compare value required to achieve the required timer output compare active/inactive delay.

**Parameters:**

- `__TIMCLK__`: timer input clock frequency (in Hz)
- `__PSC__`: prescaler
- `__DELAY__`: timer output compare active/inactive delay (in us)

**Return value:**

- Compare: value (between Min\_Data=0 and Max\_Data=65535)

**Notes:**

- ex: `__LL_TIM_CALC_DELAY (1000000, LL_TIM_GetPrescaler (), 10);`

**\_\_LL\_TIM\_CALC\_PULSE****Description:**

- HELPER macro calculating the auto-reload value to achieve the required pulse duration (when the timer operates in one pulse mode).

**Parameters:**

- `__TIMCLK__`: timer input clock frequency (in Hz)
- `__PSC__`: prescaler
- `__DELAY__`: timer output compare active/inactive delay (in us)
- `__PULSE__`: pulse duration (in us)

**Return value:**

- Auto-reload: value (between Min\_Data=0 and Max\_Data=65535)

**Notes:**

- ex: `__LL_TIM_CALC_PULSE (1000000, LL_TIM_GetPrescaler (), 10, 20);`

**\_\_LL\_TIM\_GET\_ICPSC\_RATIO****Description:**

- HELPER macro retrieving the ratio of the input

capture prescaler.

**Parameters:**

- `__ICPSC__`: This parameter can be one of the following values:
  - `LL_TIM_ICPSC_DIV1`
  - `LL_TIM_ICPSC_DIV2`
  - `LL_TIM_ICPSC_DIV4`
  - `LL_TIM_ICPSC_DIV8`

**Return value:**

- Input: capture prescaler ratio (1, 2, 4 or 8)

**Notes:**

- ex: `__LL_TIM_GET_ICPSC_RATIO`  
(`LL_TIM_IC_GetPrescaler ()`);

**Common Write and read registers Macros**

`LL_TIM_WriteReg`

**Description:**

- Write a value in TIM register.

**Parameters:**

- `__INSTANCE__`: TIM Instance
- `__REG__`: Register to be written
- `__VALUE__`: Value to be written in the register

**Return value:**

- None

`LL_TIM_ReadReg`

**Description:**

- Read a value in TIM register.

**Parameters:**

- `__INSTANCE__`: TIM Instance
- `__REG__`: Register to be read

**Return value:**

- Register: value

## 67 LL USART Generic Driver

### 67.1 USART Firmware driver registers structures

#### 67.1.1 LL\_USART\_InitTypeDef

##### Data Fields

- *uint32\_t BaudRate*
- *uint32\_t DataWidth*
- *uint32\_t StopBits*
- *uint32\_t Parity*
- *uint32\_t TransferDirection*
- *uint32\_t HardwareFlowControl*
- *uint32\_t OverSampling*

##### Field Documentation

- ***uint32\_t LL\_USART\_InitTypeDef::BaudRate***  
This field defines expected Usart communication baud rate. This feature can be modified afterwards using unitary function **LL\_USART\_SetBaudRate()**.
- ***uint32\_t LL\_USART\_InitTypeDef::DataWidth***  
Specifies the number of data bits transmitted or received in a frame. This parameter can be a value of **USART\_LL\_EC\_DATAWIDTH**. This feature can be modified afterwards using unitary function **LL\_USART\_SetDataWidth()**.
- ***uint32\_t LL\_USART\_InitTypeDef::StopBits***  
Specifies the number of stop bits transmitted. This parameter can be a value of **USART\_LL\_EC\_STOPBITS**. This feature can be modified afterwards using unitary function **LL\_USART\_SetStopBitsLength()**.
- ***uint32\_t LL\_USART\_InitTypeDef::Parity***  
Specifies the parity mode. This parameter can be a value of **USART\_LL\_EC\_PARITY**. This feature can be modified afterwards using unitary function **LL\_USART\_SetParity()**.
- ***uint32\_t LL\_USART\_InitTypeDef::TransferDirection***  
Specifies whether the Receive and/or Transmit mode is enabled or disabled. This parameter can be a value of **USART\_LL\_EC\_DIRECTION**. This feature can be modified afterwards using unitary function **LL\_USART\_SetTransferDirection()**.
- ***uint32\_t LL\_USART\_InitTypeDef::HardwareFlowControl***  
Specifies whether the hardware flow control mode is enabled or disabled. This parameter can be a value of **USART\_LL\_EC\_HWCONTROL**. This feature can be modified afterwards using unitary function **LL\_USART\_SetHWFlowCtrl()**.
- ***uint32\_t LL\_USART\_InitTypeDef::OverSampling***  
Specifies whether USART oversampling mode is 16 or 8. This parameter can be a value of **USART\_LL\_EC\_OVERSAMPLING**. This feature can be modified afterwards using unitary function **LL\_USART\_SetOverSampling()**.

#### 67.1.2 LL\_USART\_ClockInitTypeDef

##### Data Fields

- *uint32\_t ClockOutput*
- *uint32\_t ClockPolarity*
- *uint32\_t ClockPhase*
- *uint32\_t LastBitClockPulse*

**Field Documentation**

- ***uint32\_t LL\_USART\_ClockInitTypeDef::ClockOutput***  
Specifies whether the USART clock is enabled or disabled. This parameter can be a value of **USART\_LL\_EC\_CLOCK**.USART HW configuration can be modified afterwards using unitary functions **LL\_USART\_EnableSCLKOutput()** or **LL\_USART\_DisableSCLKOutput()**. For more details, refer to description of this function.
- ***uint32\_t LL\_USART\_ClockInitTypeDef::ClockPolarity***  
Specifies the steady state of the serial clock. This parameter can be a value of **USART\_LL\_EC\_POLARITY**.USART HW configuration can be modified afterwards using unitary functions **LL\_USART\_SetClockPolarity()**. For more details, refer to description of this function.
- ***uint32\_t LL\_USART\_ClockInitTypeDef::ClockPhase***  
Specifies the clock transition on which the bit capture is made. This parameter can be a value of **USART\_LL\_EC\_PHASE**.USART HW configuration can be modified afterwards using unitary functions **LL\_USART\_SetClockPhase()**. For more details, refer to description of this function.
- ***uint32\_t LL\_USART\_ClockInitTypeDef::LastBitClockPulse***  
Specifies whether the clock pulse corresponding to the last transmitted data bit (MSB) has to be output on the SCLK pin in synchronous mode. This parameter can be a value of **USART\_LL\_EC\_LASTCLKPULSE**.USART HW configuration can be modified afterwards using unitary functions **LL\_USART\_SetLastClkPulseOutput()**. For more details, refer to description of this function.

## 67.2 USART Firmware driver API description

### 67.2.1 Detailed description of functions

**LL\_USART\_Enable**

|                                                   |                                                                                   |
|---------------------------------------------------|-----------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_Enable (USART_TypeDef * USARTx)</code>        |
| Function description                              | USART Enable.                                                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                  |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 UE LL_USART_Enable</li> </ul>        |

**LL\_USART\_Disable**

|                      |                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_USART_Disable (USART_TypeDef * USARTx)</code>                                                                                                                                                        |
| Function description | USART Disable (all USART prescalers and outputs are disabled)                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                  |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>• When USART is disabled, USART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the USART is kept, but all the status</li> </ul> |

flags, in the USARTx\_SR are set to their default values.

Reference Manual to  
LL API cross  
reference:

- CR1 UE LL\_USART\_Disable

### **LL\_USART\_IsEnabled**

Function name **`_STATIC_INLINE uint32_t LL_USART_IsEnabled(  
USART_TypeDef * USARTx)`**

Function description Indicate if USART is enabled.

Parameters • **USARTx:** USART Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:  
• CR1 UE LL\_USART\_IsEnabled

### **LL\_USART\_EnableDirectionRx**

Function name **`_STATIC_INLINE void LL_USART_EnableDirectionRx(  
USART_TypeDef * USARTx)`**

Function description Receiver Enable (Receiver is enabled and begins searching for a start bit)

Parameters • **USARTx:** USART Instance

Return values • **None:**

Reference Manual to  
LL API cross  
reference:  
• CR1 RE LL\_USART\_EnableDirectionRx

### **LL\_USART\_DisableDirectionRx**

Function name **`_STATIC_INLINE void LL_USART_DisableDirectionRx(  
USART_TypeDef * USARTx)`**

Function description Receiver Disable.

Parameters • **USARTx:** USART Instance

Return values • **None:**

Reference Manual to  
LL API cross  
reference:  
• CR1 RE LL\_USART\_DisableDirectionRx

### **LL\_USART\_EnableDirectionTx**

Function name **`_STATIC_INLINE void LL_USART_EnableDirectionTx(  
USART_TypeDef * USARTx)`**

Function description Transmitter Enable.

Parameters • **USARTx:** USART Instance

---

|                                                   |                                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------|
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR1 TE LL_USART_EnableDirectionTx</li> </ul> |

### LL\_USART\_DisableDirectionTx

|                                                   |                                                                                                |
|---------------------------------------------------|------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_DisableDirectionTx(<br/>    USART_TypeDef * USARTx)</code> |
| Function description                              | Transmitter Disable.                                                                           |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                 |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR1 TE LL_USART_DisableDirectionTx</li> </ul>           |

### LL\_USART\_SetTransferDirection

|                                                   |                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_SetTransferDirection(<br/>    USART_TypeDef * USARTx, uint32_t TransferDirection)</code>                                                                                                                                                                                                                                         |
| Function description                              | Configure simultaneously enabled/disabled states of Transmitter and Receiver.                                                                                                                                                                                                                                                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> <li><b>TransferDirection:</b> This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>- LL_USART_DIRECTION_NONE</li> <li>- LL_USART_DIRECTION_RX</li> <li>- LL_USART_DIRECTION_TX</li> <li>- LL_USART_DIRECTION_TX_RX</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                       |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR1 RE LL_USART_SetTransferDirection</li> <li>CR1 TE LL_USART_SetTransferDirection</li> </ul>                                                                                                                                                                                                                                 |

### LL\_USART\_GetTransferDirection

|                                     |                                                                                                                                                                                                                                                                                                             |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                       | <code>__STATIC_INLINE uint32_t LL_USART_GetTransferDirection(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                                                                                        |
| Function description                | Return enabled/disabled states of Transmitter and Receiver.                                                                                                                                                                                                                                                 |
| Parameters                          | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                             |
| Return values                       | <ul style="list-style-type: none"> <li><b>Returned:</b> value can be one of the following values:             <ul style="list-style-type: none"> <li>- LL_USART_DIRECTION_NONE</li> <li>- LL_USART_DIRECTION_RX</li> <li>- LL_USART_DIRECTION_TX</li> <li>- LL_USART_DIRECTION_TX_RX</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross | <ul style="list-style-type: none"> <li>CR1 RE LL_USART_GetTransferDirection</li> <li>CR1 TE LL_USART_GetTransferDirection</li> </ul>                                                                                                                                                                        |

reference:

### **LL\_USART\_SetParity**

|                                                   |                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_SetParity(<br/>    USART_TypeDef * USARTx, uint32_t Parity)</code>                                                                                                                                                                                                                                             |
| Function description                              | Configure Parity (enabled/disabled and parity mode if enabled).                                                                                                                                                                                                                                                                                    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>Parity:</b> This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_PARITY_NONE</li> <li>– LL_USART_PARITY_EVEN</li> <li>– LL_USART_PARITY_ODD</li> </ul> </li> </ul>                                |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                   |
| Notes                                             | <ul style="list-style-type: none"> <li>• This function selects if hardware parity control (generation and detection) is enabled or disabled. When the parity control is enabled (Odd or Even), computed parity bit is inserted at the MSB position (9th or 8th bit depending on data width) and parity is checked on the received data.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 PS LL_USART_SetParity</li> <li>• CR1 PCE LL_USART_SetParity</li> </ul>                                                                                                                                                                                                                                |

### **LL\_USART\_GetParity**

|                                                   |                                                                                                                                                                                                                                                                     |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_GetParity(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                                                           |
| Function description                              | Return Parity configuration (enabled/disabled and parity mode if enabled)                                                                                                                                                                                           |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                   |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_PARITY_NONE</li> <li>– LL_USART_PARITY_EVEN</li> <li>– LL_USART_PARITY_ODD</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 PS LL_USART_GetParity</li> <li>• CR1 PCE LL_USART_GetParity</li> </ul>                                                                                                                                                 |

### **LL\_USART\_SetWakeUpMethod**

|                      |                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_USART_SetWakeUpMethod(<br/>    USART_TypeDef * USARTx, uint32_t Method)</code>                                                                                                                                                                                    |
| Function description | Set Receiver Wake Up method from Mute mode.                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>Method:</b> This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_WAKEUP_IDLELINE</li> <li>– LL_USART_WAKEUP_ADDRESSMARK</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                |

Reference Manual to  
LL API cross  
reference:

- CR1 WAKE LL\_USART\_SetWakeUpMethod

### **LL\_USART\_GetWakeUpMethod**

|                                                   |                                                                                                                                                                                                                                               |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_GetWakeUpMethod(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                               |
| Function description                              | Return Receiver Wake Up method from Mute mode.                                                                                                                                                                                                |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_USART_WAKEUP_IDLELINE</li> <li>– LL_USART_WAKEUP_ADDRESSMARK</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 WAKE LL_USART_GetWakeUpMethod</li> </ul>                                                                                                                                                         |

### **LL\_USART\_SetDataWidth**

|                                                   |                                                                                                                                                                                                                                                                                         |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_SetDataWidth(<br/>    USART_TypeDef * USARTx, uint32_t DataWidth)</code>                                                                                                                                                                            |
| Function description                              | Set Word length (i.e.                                                                                                                                                                                                                                                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>DataWidth:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_USART_DATAWIDTH_8B</li> <li>– LL_USART_DATAWIDTH_9B</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                        |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 M LL_USART_SetDataWidth</li> </ul>                                                                                                                                                                                                         |

### **LL\_USART\_GetDataWidth**

|                                                   |                                                                                                                                                                                                                                      |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_GetDataWidth(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                         |
| Function description                              | Return Word length (i.e.                                                                                                                                                                                                             |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                    |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_USART_DATAWIDTH_8B</li> <li>– LL_USART_DATAWIDTH_9B</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 M LL_USART_SetDataWidth</li> </ul>                                                                                                                                                      |

**LL\_USART\_SetOverSampling**

|                                                   |                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_SetOverSampling(<br/>    USART_TypeDef * USARTx, uint32_t OverSampling)</code>                                                                                                                                                                                |
| Function description                              | Set Oversampling to 8-bit or 16-bit mode.                                                                                                                                                                                                                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>OverSampling:</b> This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_OVERSAMPLING_16</li> <li>– LL_USART_OVERSAMPLING_8</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                  |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 OVER8 LL_USART_SetOverSampling</li> </ul>                                                                                                                                                                                                            |

**LL\_USART\_GetOverSampling**

|                                                   |                                                                                                                                                                                                                                             |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_GetOverSampling(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                             |
| Function description                              | Return Oversampling mode.                                                                                                                                                                                                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_OVERSAMPLING_16</li> <li>– LL_USART_OVERSAMPLING_8</li> </ul> </li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 OVER8 LL_USART_GetOverSampling</li> </ul>                                                                                                                                                      |

**LL\_USART\_SetLastClkPulseOutput**

|                                                   |                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_SetLastClkPulseOutput(<br/>    USART_TypeDef * USARTx, uint32_t LastBitClockPulse)</code>                                                                                                                                                                                      |
| Function description                              | Configure if Clock pulse of the last data bit is output to the SCLK pin or not.                                                                                                                                                                                                                                    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>LastBitClockPulse:</b> This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_LASTCLKPULSE_NO_OUTPUT</li> <li>– LL_USART_LASTCLKPULSE_OUTPUT</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                   |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li> </ul>                                                                                                                                      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LBCL LL_USART_SetLastClkPulseOutput</li> </ul>                                                                                                                                                                                                                        |

**LL\_USART\_GetLastClkPulseOutput**

|                                                   |                                                                                                                                                                                                                                                         |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t<br/>LL_USART_GetLastClkPulseOutput (USART_TypeDef *<br/>USARTx)</code>                                                                                                                                                   |
| Function description                              | Retrieve Clock pulse of the last data bit output configuration (Last bit Clock pulse output to the SCLK pin or not)                                                                                                                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                       |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_LASTCLKPULSE_NO_OUTPUT</li> <li>– LL_USART_LASTCLKPULSE_OUTPUT</li> </ul> </li> </ul> |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li> </ul>                                                                           |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LBCL LL_USART_GetLastClkPulseOutput</li> </ul>                                                                                                                                                             |

**LL\_USART\_SetClockPhase**

|                                                   |                                                                                                                                                                                                                                                                                          |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_SetClockPhase<br/>(USART_TypeDef * USARTx, uint32_t ClockPhase)</code>                                                                                                                                                                               |
| Function description                              | Select the phase of the clock output on the SCLK pin in synchronous mode.                                                                                                                                                                                                                |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>ClockPhase:</b> This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_PHASE_1EDGE</li> <li>– LL_USART_PHASE_2EDGE</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                         |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li> </ul>                                                                                                            |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 CPHA LL_USART_SetClockPhase</li> </ul>                                                                                                                                                                                                      |

**LL\_USART\_GetClockPhase**

|                      |                                                                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_USART_GetClockPhase<br/>(USART_TypeDef * USARTx)</code>                                                                                                                                            |
| Function description | Return phase of the clock output on the SCLK pin in synchronous mode.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                    |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:             <ul style="list-style-type: none"> <li>– LL_USART_PHASE_1EDGE</li> <li>– LL_USART_PHASE_2EDGE</li> </ul> </li> </ul> |

- Notes
- Macro IS\_USART\_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

- Reference Manual to LL API cross reference:
- CR2 CPHA LL\_USART\_GetClockPhase

### LL\_USART\_SetClockPolarity

|                                             |                                                                                                                                                                                                                                                                                          |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE void LL_USART_SetClockPolarity(<br/>    USART_TypeDef * USARTx, uint32_t ClockPolarity)</code>                                                                                                                                                                     |
| Function description                        | Select the polarity of the clock output on the SCLK pin in synchronous mode.                                                                                                                                                                                                             |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> <li><b>ClockPolarity:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_USART_POLARITY_LOW</li> <li>– LL_USART_POLARITY_HIGH</li> </ul> </li> </ul> |
| Return values                               | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                                                                                                                                           |
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li> </ul>                                                                                                              |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CR2 CPOL LL_USART_SetClockPolarity</li> </ul>                                                                                                                                                                                                     |

### LL\_USART\_GetClockPolarity

|                                             |                                                                                                                                                                                                                                     |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE uint32_t LL_USART_GetClockPolarity(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                    |
| Function description                        | Return polarity of the clock output on the SCLK pin in synchronous mode.                                                                                                                                                            |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                     |
| Return values                               | <ul style="list-style-type: none"> <li><b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_USART_POLARITY_LOW</li> <li>– LL_USART_POLARITY_HIGH</li> </ul> </li> </ul> |
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li> </ul>                                                         |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CR2 CPOL LL_USART_GetClockPolarity</li> </ul>                                                                                                                                                |

### LL\_USART\_ConfigClock

|               |                                                                                                                                                         |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name | <code>__STATIC_INLINE void LL_USART_ConfigClock(<br/>    USART_TypeDef * USARTx, uint32_t Phase, uint32_t Polarity,<br/>    uint32_t LBCPOutput)</code> |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|

|                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function description                        | Configure Clock signal format (Phase Polarity and choice about output of last bit clock pulse)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>Phase:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_USART_PHASE_1EDGE</li> <li>- LL_USART_PHASE_2EDGE</li> </ul> </li> <li>• <b>Polarity:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_USART_POLARITY_LOW</li> <li>- LL_USART_POLARITY_HIGH</li> </ul> </li> <li>• <b>LBCPOutput:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- LL_USART_LASTCLKPULSE_NO_OUTPUT</li> <li>- LL_USART_LASTCLKPULSE_OUTPUT</li> </ul> </li> </ul> |
| Return values                               | • <b>None:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                                       | <ul style="list-style-type: none"> <li>• Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li> <li>• Call of this function is equivalent to following function call sequence : Clock Phase configuration using LL_USART_SetClockPhase() functionClock Polarity configuration using LL_USART_SetClockPolarity() functionOutput of Last bit Clock pulse configuration using LL_USART_SetLastClkPulseOutput() function</li> </ul>                                                                                                                                                                           |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CR2 CPHA LL_USART_ConfigClock</li> <li>• CR2 CPOL LL_USART_ConfigClock</li> <li>• CR2 LBCL LL_USART_ConfigClock</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

### LL\_USART\_EnableSCLKOutput

|                                             |                                                                                                                                                                               |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <b>STATIC_INLINE void LL_USART_EnableSCLKOutput (USART_TypeDef * USARTx)</b>                                                                                                  |
| Function description                        | Enable Clock output on SCLK pin.                                                                                                                                              |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                             |
| Return values                               | • <b>None:</b>                                                                                                                                                                |
| Notes                                       | <ul style="list-style-type: none"> <li>• Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CR2 CLKEN LL_USART_EnableSCLKOutput</li> </ul>                                                                                       |

### LL\_USART\_DisableSCLKOutput

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| Function name        | <b>STATIC_INLINE void LL_USART_DisableSCLKOutput (USART_TypeDef * USARTx)</b>     |
| Function description | Disable Clock output on SCLK pin.                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul> |

---

|                                             |                                                                                                                                                                           |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Return values                               | <ul style="list-style-type: none"><li><b>None:</b></li></ul>                                                                                                              |
| Notes                                       | <ul style="list-style-type: none"><li>Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li></ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"><li>CR2 CLKEN LL_USART_DisableSCLKOutput</li></ul>                                                                                      |

### LL\_USART\_IsEnabledSCLKOutput

|                                             |                                                                                                                                                                           |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledSCLKOutput(<br/>    USART_TypeDef * USARTx)</code>                                                                       |
| Function description                        | Indicate if Clock output on SCLK pin is enabled.                                                                                                                          |
| Parameters                                  | <ul style="list-style-type: none"><li><b>USARTx:</b> USART Instance</li></ul>                                                                                             |
| Return values                               | <ul style="list-style-type: none"><li><b>State:</b> of bit (1 or 0).</li></ul>                                                                                            |
| Notes                                       | <ul style="list-style-type: none"><li>Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.</li></ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"><li>CR2 CLKEN LL_USART_IsEnabledSCLKOutput</li></ul>                                                                                    |

### LL\_USART\_SetStopBitsLength

|                                             |                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE void LL_USART_SetStopBitsLength(<br/>    USART_TypeDef * USARTx, uint32_t StopBits)</code>                                                                                                                                                                                                             |
| Function description                        | Set the length of the stop bits.                                                                                                                                                                                                                                                                                             |
| Parameters                                  | <ul style="list-style-type: none"><li><b>USARTx:</b> USART Instance</li><li><b>StopBits:</b> This parameter can be one of the following values:<ul style="list-style-type: none"><li>– LL_USART_STOPBITS_0_5</li><li>– LL_USART_STOPBITS_1</li><li>– LL_USART_STOPBITS_1_5</li><li>– LL_USART_STOPBITS_2</li></ul></li></ul> |
| Return values                               | <ul style="list-style-type: none"><li><b>None:</b></li></ul>                                                                                                                                                                                                                                                                 |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"><li>CR2 STOP LL_USART_SetStopBitsLength</li></ul>                                                                                                                                                                                                                                          |

### LL\_USART\_GetStopBitsLength

|                      |                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_USART_GetStopBitsLength(<br/>    USART_TypeDef * USARTx)</code>                                                                                                               |
| Function description | Retrieve the length of the stop bits.                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"><li><b>USARTx:</b> USART Instance</li></ul>                                                                                                                                   |
| Return values        | <ul style="list-style-type: none"><li><b>Returned:</b> value can be one of the following values:<ul style="list-style-type: none"><li>– LL_USART_STOPBITS_0_5</li><li>– LL_USART_STOPBITS_1</li></ul></li></ul> |

Reference Manual to  
LL API cross  
reference:

- LL\_USART\_STOPBITS\_1\_5
- LL\_USART\_STOPBITS\_2
- CR2 STOP LL\_USART\_GetStopBitsLength

### LL\_USART\_ConfigCharacter

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_ConfigCharacter(<br/>    USART_TypeDef * USARTx, uint32_t DataWidth, uint32_t<br/>    Parity, uint32_t StopBits)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Function description                              | Configure Character frame format (Datawidth, Parity control, Stop Bits)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>DataWidth:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_USART_DATAWIDTH_8B</li> <li>- LL_USART_DATAWIDTH_9B</li> </ul> </li> <li>• <b>Parity:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_USART_PARITY_NONE</li> <li>- LL_USART_PARITY_EVEN</li> <li>- LL_USART_PARITY_ODD</li> </ul> </li> <li>• <b>StopBits:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_USART_STOPBITS_0_5</li> <li>- LL_USART_STOPBITS_1</li> <li>- LL_USART_STOPBITS_1_5</li> <li>- LL_USART_STOPBITS_2</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                                             | <ul style="list-style-type: none"> <li>• Call of this function is equivalent to following function call sequence : Data Width configuration using LL_USART_SetDataWidth() functionParity Control and mode configuration using LL_USART_SetParity() functionStop bits configuration using LL_USART_SetStopBitsLength() function</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 PS LL_USART_ConfigCharacter</li> <li>• CR1 PCE LL_USART_ConfigCharacter</li> <li>• CR1 M LL_USART_ConfigCharacter</li> <li>• CR2 STOP LL_USART_ConfigCharacter</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

### LL\_USART\_SetNodeAddress

|                      |                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_USART_SetNodeAddress(<br/>    USART_TypeDef * USARTx, uint32_t NodeAddress)</code>                                              |
| Function description | Set Address of the USART node.                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>NodeAddress:</b> 4 bit Address of the USART node.</li> </ul>             |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• This is used in multiprocessor communication during Mute mode or Stop mode, for wake up with address mark</li> </ul> |

- detection.
- CR2 ADD LL\_USART\_SetNodeAddress
- Reference Manual to  
LL API cross  
reference:

### **LL\_USART\_GetNodeAddress**

|                                                   |                                                                                                                                   |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE uint32_t LL_USART_GetNodeAddress(<br/>USART_TypeDef * USARTx)</code></b>                                  |
| Function description                              | Return 4 bit Address of the USART node as set in ADD field of CR2.                                                                |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                 |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Address:</b> of the USART node (Value between Min_Data=0 and Max_Data=255)</li> </ul> |
| Notes                                             | <ul style="list-style-type: none"> <li>• only 4bits (b3-b0) of returned value are relevant (b31-b4 are not relevant)</li> </ul>   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 ADD LL_USART_GetNodeAddress</li> </ul>                                               |

### **LL\_USART\_EnableRTSHWFlowCtrl**

|                                                   |                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_EnableRTSHWFlowCtrl(<br/>USART_TypeDef * USARTx)</code></b>                                                                                                |
| Function description                              | Enable RTS HW Flow Control.                                                                                                                                                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                 |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 RTSE LL_USART_EnableRTSHWFlowCtrl</li> </ul>                                                                                                        |

### **LL\_USART\_DisableRTSHWFlowCtrl**

|                                                   |                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_DisableRTSHWFlowCtrl(<br/>USART_TypeDef * USARTx)</code></b>                                                                                               |
| Function description                              | Disable RTS HW Flow Control.                                                                                                                                                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                 |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 RTSE LL_USART_DisableRTSHWFlowCtrl</li> </ul>                                                                                                       |

reference:

### **LL\_USART\_EnableCTSHWFlowCtrl**

|                                                   |                                                                                                                                                                                                               |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_EnableCTSHWFlowCtrl(USART_TypeDef * USARTx)</code></b>                                                                                                                  |
| Function description                              | Enable CTS HW Flow Control.                                                                                                                                                                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                              |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro <code>IS_UART_HWFLOW_INSTANCE(USARTx)</code> can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 CTSE LL_USART_EnableCTSHWFlowCtrl</li> </ul>                                                                                                                     |

### **LL\_USART\_DisableCTSHWFlowCtrl**

|                                                   |                                                                                                                                                                                                               |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_DisableCTSHWFlowCtrl(USART_TypeDef * USARTx)</code></b>                                                                                                                 |
| Function description                              | Disable CTS HW Flow Control.                                                                                                                                                                                  |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                              |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro <code>IS_UART_HWFLOW_INSTANCE(USARTx)</code> can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 CTSE LL_USART_DisableCTSHWFlowCtrl</li> </ul>                                                                                                                    |

### **LL\_USART\_SetHWFlowCtrl**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE void LL_USART_SetHWFlowCtrl(USART_TypeDef * USARTx, uint32_t HardwareFlowControl)</code></b>                                                                                                                                                                                                                                                                                                   |
| Function description | Configure HW Flow Control mode (both CTS and RTS)                                                                                                                                                                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>HardwareFlowControl:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <code>LL_USART_HWCONTROL_NONE</code></li> <li>- <code>LL_USART_HWCONTROL_RTS</code></li> <li>- <code>LL_USART_HWCONTROL_CTS</code></li> <li>- <code>LL_USART_HWCONTROL_RTS_CTS</code></li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• Macro <code>IS_UART_HWFLOW_INSTANCE(USARTx)</code> can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul>                                                                                                                                                                                                          |

- Reference Manual to  
LL API cross  
reference:
- CR3 RTSE LL\_USART\_SetHWFlowCtrl
  - CR3 CTSE LL\_USART\_SetHWFlowCtrl

### **LL\_USART\_GetHWFlowCtrl**

|                                                   |                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_GetHWFlowCtrl(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                                                                                                   |
| Function description                              | Return HW Flow Control configuration (both CTS and RTS)                                                                                                                                                                                                                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                               |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>- LL_USART_HWCONTROL_NONE</li> <li>- LL_USART_HWCONTROL_RTS</li> <li>- LL_USART_HWCONTROL_CTS</li> <li>- LL_USART_HWCONTROL_RTS_CTS</li> </ul> </li> </ul> |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul>                                                                                                                |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 RTSE LL_USART_GetHWFlowCtrl</li> <li>• CR3 CTSE LL_USART_GetHWFlowCtrl</li> </ul>                                                                                                                                                                                  |

### **LL\_USART\_EnableOneBitSamp**

|                                                   |                                                                                              |
|---------------------------------------------------|----------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_EnableOneBitSamp(<br/>    USART_TypeDef * USARTx)</code> |
| Function description                              | Enable One bit sampling method.                                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                             |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 ONEBIT LL_USART_EnableOneBitSamp</li> </ul>     |

### **LL\_USART\_DisableOneBitSamp**

|                                                   |                                                                                               |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_DisableOneBitSamp(<br/>    USART_TypeDef * USARTx)</code> |
| Function description                              | Disable One bit sampling method.                                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                              |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 ONEBIT LL_USART_DisableOneBitSamp</li> </ul>     |

### **LL\_USART\_IsEnabledOneBitSamp**

|               |                                                                    |
|---------------|--------------------------------------------------------------------|
| Function name | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledOneBitSamp</code> |
|---------------|--------------------------------------------------------------------|

**(USART\_TypeDef \* USARTx)**

|                                                   |                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------|
| Function description                              | Indicate if One bit sampling method is enabled.                                    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>  |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | CR3 ONEBIT LL_USART_IsEnabledOneBitSamp                                            |

**LL\_USART\_SetBaudRate**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b>__STATIC_INLINE void LL_USART_SetBaudRate<br/>(USART_TypeDef * USARTx, uint32_t PeriphClk, uint32_t<br/>OverSampling, uint32_t BaudRate)</b>                                                                                                                                                                                                                           |
| Function description                              | Configure USART BRR register for achieving expected Baud Rate value.                                                                                                                                                                                                                                                                                                      |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>PeriphClk:</b> Peripheral Clock</li> <li>• <b>OverSampling:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– LL_USART_OVERSAMPLING_16</li> <li>– LL_USART_OVERSAMPLING_8</li> </ul> </li> <li>• <b>BaudRate:</b> Baud Rate</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                          |
| Notes                                             | <ul style="list-style-type: none"> <li>• Compute and set USARTDIV value in BRR Register (full BRR content) according to used Peripheral Clock, Oversampling mode, and expected Baud Rate values</li> <li>• Peripheral clock and Baud rate values provided as function parameters should be valid (Baud rate value != 0)</li> </ul>                                        |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• BRR BRR LL_USART_SetBaudRate</li> </ul>                                                                                                                                                                                                                                                                                          |

**LL\_USART\_GetBaudRate**

|                      |                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>__STATIC_INLINE uint32_t LL_USART_GetBaudRate<br/>(USART_TypeDef * USARTx, uint32_t PeriphClk, uint32_t<br/>OverSampling)</b>                                                                                                                                                                                                    |
| Function description | Return current Baud Rate value, according to USARTDIV present in BRR register (full BRR content), and to used Peripheral Clock and Oversampling mode values.                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>PeriphClk:</b> Peripheral Clock</li> <li>• <b>OverSampling:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– LL_USART_OVERSAMPLING_16</li> <li>– LL_USART_OVERSAMPLING_8</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Baud:</b> Rate</li> </ul>                                                                                                                                                                                                                                                               |

---

|                                             |                                                                                                                                                 |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Notes                                       | <ul style="list-style-type: none"> <li>In case of non-initialized or invalid value stored in BRR register, value 0 will be returned.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>BRR BRR LL_USART_GetBaudRate</li> </ul>                                                                  |

### LL\_USART\_EnableIrda

|                                             |                                                                                                                                                                        |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <b><code>STATIC_INLINE void LL_USART_EnableIrda(USART_TypeDef * USARTx)</code></b>                                                                                     |
| Function description                        | Enable IrDA mode.                                                                                                                                                      |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                        |
| Return values                               | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                         |
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CR3 IREN LL_USART_EnableIrda</li> </ul>                                                                                         |

### LL\_USART\_DisableIrda

|                                             |                                                                                                                                                                        |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <b><code>STATIC_INLINE void LL_USART_DisableIrda(USART_TypeDef * USARTx)</code></b>                                                                                    |
| Function description                        | Disable IrDA mode.                                                                                                                                                     |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                        |
| Return values                               | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                         |
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CR3 IREN LL_USART_DisableIrda</li> </ul>                                                                                        |

### LL\_USART\_IsEnabledIrda

|                                  |                                                                                                                                                                        |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                    | <b><code>STATIC_INLINE uint32_t LL_USART_IsEnabledIrda(USART_TypeDef * USARTx)</code></b>                                                                              |
| Function description             | Indicate if IrDA mode is enabled.                                                                                                                                      |
| Parameters                       | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                        |
| Return values                    | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>                                                                                       |
| Notes                            | <ul style="list-style-type: none"> <li>Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to LL API cross | <ul style="list-style-type: none"> <li>CR3 IREN LL_USART_IsEnabledIrda</li> </ul>                                                                                      |

reference:

### **LL\_USART\_SetIrdaPowerMode**

|                                                   |                                                                                                                                                                                                                                                                                                                |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>STATIC_INLINE void LL_USART_SetIrdaPowerMode(USART_TypeDef * USARTx, uint32_t PowerMode)</code></b>                                                                                                                                                                                                   |
| Function description                              | Configure IrDA Power Mode (Normal or Low Power)                                                                                                                                                                                                                                                                |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>PowerMode:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <code>LL_USART_IRDA_POWER_NORMAL</code></li> <li>- <code>LL_USART_IRDA_POWER_LOW</code></li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                               |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro <code>IS_IRDA_INSTANCE(USARTx)</code> can be used to check whether or not IrDA feature is supported by the USARTx instance.</li> </ul>                                                                                                                          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 IRLP <code>LL_USART_SetIrdaPowerMode</code></li> </ul>                                                                                                                                                                                                            |

### **LL\_USART\_GetIrdaPowerMode**

|                                                   |                                                                                                                                                                                                                                                          |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>STATIC_INLINE uint32_t LL_USART_GetIrdaPowerMode(USART_TypeDef * USARTx)</code></b>                                                                                                                                                             |
| Function description                              | Retrieve IrDA Power Mode configuration (Normal or Low Power)                                                                                                                                                                                             |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                        |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values: <ul style="list-style-type: none"> <li>- <code>LL_USART_IRDA_POWER_NORMAL</code></li> <li>- <code>LL_USART_PHASE_2EDGE</code></li> </ul> </li> </ul> |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro <code>IS_IRDA_INSTANCE(USARTx)</code> can be used to check whether or not IrDA feature is supported by the USARTx instance.</li> </ul>                                                                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 IRLP <code>LL_USART_GetIrdaPowerMode</code></li> </ul>                                                                                                                                                      |

### **LL\_USART\_SetIrdaPrescaler**

|                      |                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>STATIC_INLINE void LL_USART_SetIrdaPrescaler(USART_TypeDef * USARTx, uint32_t PrescalerValue)</code></b>                                                 |
| Function description | Set Irda prescaler value, used for dividing the USART clock source to achieve the Irda Low Power frequency (8 bits value)                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>PrescalerValue:</b> Value between Min_Data=0x00 and Max_Data=0xFF</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                  |

---

|                                             |                                                                                                                                                                        |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>GTPR PSC LL_USART_SetIrdaPrescaler</li> </ul>                                                                                   |

### LL\_USART\_GetIrdaPrescaler

|                                             |                                                                                                                                                                        |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE uint32_t LL_USART_GetIrdaPrescaler(<br/>          USART_TypeDef * USARTx)</code>                                                                 |
| Function description                        | Return Irda prescaler value, used for dividing the USART clock source to achieve the Irda Low Power frequency (8 bits value)                                           |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                        |
| Return values                               | <ul style="list-style-type: none"> <li><b>Irda:</b> prescaler value (Value between Min_Data=0x00 and Max_Data=0xFF)</li> </ul>                                         |
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>GTPR PSC LL_USART_GetIrdaPrescaler</li> </ul>                                                                                   |

### LL\_USART\_EnableSmartcardNACK

|                                             |                                                                                                                                                                                  |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE void LL_USART_EnableSmartcardNACK(<br/>          USART_TypeDef * USARTx)</code>                                                                            |
| Function description                        | Enable Smartcard NACK transmission.                                                                                                                                              |
| Parameters                                  | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                                  |
| Return values                               | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                                   |
| Notes                                       | <ul style="list-style-type: none"> <li>Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>CR3 NACK LL_USART_EnableSmartcardNACK</li> </ul>                                                                                          |

### LL\_USART\_DisableSmartcardNACK

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_USART_DisableSmartcardNACK(<br/>          USART_TypeDef * USARTx)</code>                                                          |
| Function description | Disable Smartcard NACK transmission.                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                 |
| Return values        | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the</li> </ul> |

USARTx instance.

- Reference Manual to LL API cross reference:
- CR3 NACK LL\_USART\_DisableSmartcardNACK

### **LL\_USART\_IsEnabledSmartcardNACK**

|                                             |                                                                                                                                                                                      |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledSmartcardNACK (USART_TypeDef * USARTTx)</code>                                                                                      |
| Function description                        | Indicate if Smartcard NACK transmission is enabled.                                                                                                                                  |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                                                                                                   |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                                   |
| Notes                                       | <ul style="list-style-type: none"> <li>• Macro IS_SMARTCARD_INSTANCE(USARTTx) can be used to check whether or not Smartcard feature is supported by the USARTTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CR3 NACK LL_USART_IsEnabledSmartcardNACK</li> </ul>                                                                                         |

### **LL\_USART\_EnableSmartcard**

|                                             |                                                                                                                                                                                      |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE void LL_USART_EnableSmartcard (USART_TypeDef * USARTTx)</code>                                                                                                 |
| Function description                        | Enable Smartcard mode.                                                                                                                                                               |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                                                                                                   |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                     |
| Notes                                       | <ul style="list-style-type: none"> <li>• Macro IS_SMARTCARD_INSTANCE(USARTTx) can be used to check whether or not Smartcard feature is supported by the USARTTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CR3 SCEN LL_USART_EnableSmartcard</li> </ul>                                                                                                |

### **LL\_USART\_DisableSmartcard**

|                                  |                                                                                                                                                                                      |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                    | <code>__STATIC_INLINE void LL_USART_DisableSmartcard (USART_TypeDef * USARTTx)</code>                                                                                                |
| Function description             | Disable Smartcard mode.                                                                                                                                                              |
| Parameters                       | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                                                                                                   |
| Return values                    | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                     |
| Notes                            | <ul style="list-style-type: none"> <li>• Macro IS_SMARTCARD_INSTANCE(USARTTx) can be used to check whether or not Smartcard feature is supported by the USARTTx instance.</li> </ul> |
| Reference Manual to LL API cross | <ul style="list-style-type: none"> <li>• CR3 SCEN LL_USART_DisableSmartcard</li> </ul>                                                                                               |

reference:

### **LL\_USART\_IsEnabledSmartcard**

|                                                   |                                                                                                                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE uint32_t LL_USART_IsEnabledSmartcard(USART_TypeDef * USARTx)</code></b>                                                                                    |
| Function description                              | Indicate if Smartcard mode is enabled.                                                                                                                                             |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                  |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                                 |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 SCEN LL_USART_IsEnabledSmartcard</li> </ul>                                                                                           |

### **LL\_USART\_SetSmartcardPrescaler**

|                                                   |                                                                                                                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_SetSmartcardPrescaler(USART_TypeDef * USARTx, uint32_t PrescalerValue)</code></b>                                                            |
| Function description                              | Set Smartcard prescaler value, used for dividing the USART clock source to provide the SMARTCARD Clock (5 bits value)                                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>PrescalerValue:</b> Value between Min_Data=0 and Max_Data=31</li> </ul>                       |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                   |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• GTPR PSC LL_USART_SetSmartcardPrescaler</li> </ul>                                                                                        |

### **LL\_USART\_GetSmartcardPrescaler**

|                      |                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE uint32_t LL_USART_GetSmartcardPrescaler(USART_TypeDef * USARTx)</code></b>                                                                                 |
| Function description | Return Smartcard prescaler value, used for dividing the USART clock source to provide the SMARTCARD Clock (5 bits value)                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                  |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Smartcard:</b> prescaler value (Value between Min_Data=0 and Max_Data=31)</li> </ul>                                                   |
| Notes                | <ul style="list-style-type: none"> <li>• Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.</li> </ul> |

- Reference Manual to LL API cross reference:
- GTPR PSC LL\_USART\_SetSmartcardPrescaler

### LL\_USART\_SetSmartcardGuardTime

|                                             |                                                                                                                                                                                      |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE void LL_USART_SetSmartcardGuardTime (USART_TypeDef * USARTx, uint32_t GuardTime)</code>                                                                        |
| Function description                        | Set Smartcard Guard time value, expressed in nb of baud clocks periods (GT[7:0] bits : Guard time value)                                                                             |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>GuardTime:</b> Value between Min_Data=0x00 and Max_Data=0xFF</li> <li>• <b>None:</b></li> </ul> |
| Return values                               |                                                                                                                                                                                      |
| Notes                                       | <ul style="list-style-type: none"> <li>• Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.</li> </ul>   |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• GTPR GT LL_USART_SetSmartcardGuardTime</li> </ul>                                                                                           |

### LL\_USART\_GetSmartcardGuardTime

|                                             |                                                                                                                                                                                    |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE uint32_t LL_USART_GetSmartcardGuardTime (USART_TypeDef * USARTx)</code>                                                                                      |
| Function description                        | Return Smartcard Guard time value, expressed in nb of baud clocks periods (GT[7:0] bits : Guard time value)                                                                        |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                  |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>Smartcard:</b> Guard time value (Value between Min_Data=0x00 and Max_Data=0xFF)</li> </ul>                                             |
| Notes                                       | <ul style="list-style-type: none"> <li>• Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• GTPR GT LL_USART_GetSmartcardGuardTime</li> </ul>                                                                                         |

### LL\_USART\_EnableHalfDuplex

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_USART_EnableHalfDuplex (USART_TypeDef * USARTx)</code>                                                                  |
| Function description | Enable Single Wire Half-Duplex mode.                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                     |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• Macro IS_UART_HALFDUPLEX_INSTANCE(USARTx) can be used to check whether or not Half-Duplex mode is</li> </ul> |

supported by the USARTx instance.

Reference Manual to  
LL API cross  
reference:

- CR3 HDSEL LL\_USART\_EnableHalfDuplex

### **LL\_USART\_DisableHalfDuplex**

|                                                   |                                                                                                                                                                                         |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE void LL_USART_DisableHalfDuplex(<br/>USART_TypeDef * USARTx)</code></b>                                                                                        |
| Function description                              | Disable Single Wire Half-Duplex mode.                                                                                                                                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                       |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                        |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_HALFDUPLEX_INSTANCE(USARTx) can be used to check whether or not Half-Duplex mode is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 HDSEL LL_USART_DisableHalfDuplex</li> </ul>                                                                                                |

### **LL\_USART\_IsEnabledHalfDuplex**

|                                                   |                                                                                                                                                                                         |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE uint32_t LL_USART_IsEnabledHalfDuplex(<br/>USART_TypeDef * USARTx)</code></b>                                                                                  |
| Function description                              | Indicate if Single Wire Half-Duplex mode is enabled.                                                                                                                                    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                       |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                                      |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_HALFDUPLEX_INSTANCE(USARTx) can be used to check whether or not Half-Duplex mode is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 HDSEL LL_USART_IsEnabledHalfDuplex</li> </ul>                                                                                              |

### **LL\_USART\_SetLINBrkDetectionLen**

|                      |                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE void LL_USART_SetLINBrkDetectionLen(<br/>USART_TypeDef * USARTx, uint32_t LINBDLength)</code></b>                                                                                                                                                                                              |
| Function description | Set LIN Break Detection Length.                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> <li>• <b>LINBDLength:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <code>LL_USART_LINBREAK_DETECT_10B</code></li> <li>– <code>LL_USART_LINBREAK_DETECT_11B</code></li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx</li> </ul>                                                                                                                                                       |

instance.

Reference Manual to  
LL API cross  
reference:

- CR2 LBDL LL\_USART\_SetLINBrkDetectionLen

### **LL\_USART\_GetLINBrkDetectionLen**

|                                                   |                                                                                                                                                                                                                                                    |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t<br/>LL_USART_GetLINBrkDetectionLen (USART_TypeDef *<br/>USARTx)</code>                                                                                                                                              |
| Function description                              | Return LIN Break Detection Length.                                                                                                                                                                                                                 |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                  |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_USART_LINBREAK_DETECT_10B</li> <li>– LL_USART_LINBREAK_DETECT_11B</li> </ul> </li> </ul> |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.</li> </ul>                                                                        |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LBDL LL_USART_GetLINBrkDetectionLen</li> </ul>                                                                                                                                                        |

### **LL\_USART\_EnableLIN**

|                                                   |                                                                                                                                                                             |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_EnableLIN<br/>(USART_TypeDef * USARTx)</code>                                                                                           |
| Function description                              | Enable LIN mode.                                                                                                                                                            |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                            |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LINEN LL_USART_EnableLIN</li> </ul>                                                                                            |

### **LL\_USART\_DisableLIN**

|                      |                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_USART_DisableLIN<br/>(USART_TypeDef * USARTx)</code>                                                                                          |
| Function description | Disable LIN mode.                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                           |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.</li> </ul> |

|                                                   |                                                                                   |
|---------------------------------------------------|-----------------------------------------------------------------------------------|
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LINEN LL_USART_DisableLIN</li> </ul> |
|---------------------------------------------------|-----------------------------------------------------------------------------------|

### LL\_USART\_IsEnabledLIN

|                                                   |                                                                                                                                                                             |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledLIN(<br/>  USART_TypeDef * USARTx&gt;)</code>                                                                              |
| Function description                              | Indicate if LIN mode is enabled.                                                                                                                                            |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                          |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LINEN LL_USART_IsEnabledLIN</li> </ul>                                                                                         |

### LL\_USART\_ConfigAsyncMode

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_ConfigAsyncMode(<br/>  USART_TypeDef * USARTx)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Function description                              | Perform basic configuration of USART for enabling use in Asynchronous Mode (UART)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Notes                                             | <ul style="list-style-type: none"> <li>• In UART mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,CLKEN bit in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register.</li> <li>• Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear CLKEN in CR2 using LL_USART_DisableSCLKOutput() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() function</li> <li>• Other remaining configurations items related to Asynchronous Mode (as Baud Rate, Word length, Parity, ...) should be set using dedicated functions</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LINEN LL_USART_ConfigAsyncMode</li> <li>• CR2 CLKEN LL_USART_ConfigAsyncMode</li> <li>• CR3 SCEN LL_USART_ConfigAsyncMode</li> <li>• CR3 IREN LL_USART_ConfigAsyncMode</li> <li>• CR3 HDSEL LL_USART_ConfigAsyncMode</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**LL\_USART\_ConfigSyncMode**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE void LL_USART_ConfigSyncMode(<br/>    USART_TypeDef * USARTx)</code></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Function description                              | Perform basic configuration of USART for enabling use in Synchronous Mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                                             | <ul style="list-style-type: none"> <li>• In Synchronous mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register. This function also sets the USART in Synchronous mode.</li> <li>• Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTTx instance.</li> <li>• Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() functionSet CLKEN in CR2 using LL_USART_EnableSCLKOutput() function</li> <li>• Other remaining configurations items related to Synchronous Mode (as Baud Rate, Word length, Parity, Clock Polarity, ...) should be set using dedicated functions</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LINEN LL_USART_ConfigSyncMode</li> <li>• CR2 CLKEN LL_USART_ConfigSyncMode</li> <li>• CR3 SCEN LL_USART_ConfigSyncMode</li> <li>• CR3 IREN LL_USART_ConfigSyncMode</li> <li>• CR3 HDSEL LL_USART_ConfigSyncMode</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**LL\_USART\_ConfigLINMode**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE void LL_USART_ConfigLINMode(<br/>    USART_TypeDef * USARTx)</code></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Function description | Perform basic configuration of USART for enabling use in LIN Mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• In LIN mode, the following bits must be kept cleared: STOP and CLKEN bits in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register. This function also set the UART/USART in LIN mode.</li> <li>• Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTTx instance.</li> <li>• Call of this function is equivalent to following function call sequence : Clear CLKEN in CR2 using</li> </ul> |

|  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>LL_USART_DisableSCLKOutput() functionClear STOP in CR2 using LL_USART_SetStopBitsLength() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() functionSet LINEN in CR2 using LL_USART_EnableLIN() function</p> <ul style="list-style-type: none"> <li>• Other remaining configurations items related to LIN Mode (as Baud Rate, Word length, LIN Break Detection Length, ...) should be set using dedicated functions</li> </ul> <p>Reference Manual to LL API cross reference:</p> <ul style="list-style-type: none"> <li>• CR2 CLKEN LL_USART_ConfigLINMode</li> <li>• CR2 STOP LL_USART_ConfigLINMode</li> <li>• CR2 LINEN LL_USART_ConfigLINMode</li> <li>• CR3 IREN LL_USART_ConfigLINMode</li> <li>• CR3 SCEN LL_USART_ConfigLINMode</li> <li>• CR3 HDSEL LL_USART_ConfigLINMode</li> </ul> |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### LL\_USART\_ConfigHalfDuplexMode

|                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                               | <code>__STATIC_INLINE void LL_USART_ConfigHalfDuplexMode(<br/>                  USART_TypeDef * USARTx)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Function description                        | Perform basic configuration of USART for enabling use in Half Duplex Mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Parameters                                  | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Return values                               | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Notes                                       | <ul style="list-style-type: none"> <li>• In Half Duplex mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,CLKEN bit in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register, This function also sets the USART/USART in Half Duplex mode.</li> <li>• Macro IS_UART_HALFDUPLEX_INSTANCE(USARTx) can be used to check whether or not Half-Duplex mode is supported by the USARTx instance.</li> <li>• Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear CLKEN in CR2 using LL_USART_DisableSCLKOutput() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear IREN in CR3 using LL_USART_DisableIrda() functionSet HDSEL in CR3 using LL_USART_EnableHalfDuplex() function</li> <li>• Other remaining configurations items related to Half Duplex Mode (as Baud Rate, Word length, Parity, ...) should be set using dedicated functions</li> </ul> |
| Reference Manual to LL API cross reference: | <ul style="list-style-type: none"> <li>• CR2 LINEN LL_USART_ConfigHalfDuplexMode</li> <li>• CR2 CLKEN LL_USART_ConfigHalfDuplexMode</li> <li>• CR3 HDSEL LL_USART_ConfigHalfDuplexMode</li> <li>• CR3 SCEN LL_USART_ConfigHalfDuplexMode</li> <li>• CR3 IREN LL_USART_ConfigHalfDuplexMode</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

**LL\_USART\_ConfigSmartcardMode**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_ConfigSmartcardMode(<br/>USART_TypeDef * USARTx)</code></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Function description                              | Perform basic configuration of USART for enabling use in Smartcard Mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Notes                                             | <ul style="list-style-type: none"> <li>• In Smartcard mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register. This function also configures Stop bits to 1.5 bits and sets the USART in Smartcard mode (SCEN bit). Clock Output is also enabled (CLKEN).</li> <li>• Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.</li> <li>• Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() functionConfigure STOP in CR2 using LL_USART_SetStopBitsLength() functionSet CLKEN in CR2 using LL_USART_EnableSCLKOutput() functionSet SCEN in CR3 using LL_USART_EnableSmartcard() function</li> <li>• Other remaining configurations items related to Smartcard Mode (as Baud Rate, Word length, Parity, ...) should be set using dedicated functions</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LINEN LL_USART_ConfigSmartcardMode</li> <li>• CR2 STOP LL_USART_ConfigSmartcardMode</li> <li>• CR2 CLKEN LL_USART_ConfigSmartcardMode</li> <li>• CR3 HDSEL LL_USART_ConfigSmartcardMode</li> <li>• CR3 SCEN LL_USART_ConfigSmartcardMode</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

**LL\_USART\_ConfigIrdaMode**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE void LL_USART_ConfigIrdaMode(<br/>USART_TypeDef * USARTx)</code></b>                                                                                                                                                                                                                                                                                                                                                               |
| Function description | Perform basic configuration of USART for enabling use in Irda Mode.                                                                                                                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                                                                                                                                                                         |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• In IRDA mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,STOP and CLKEN bits in the USART_CR2 register,SCEN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register. This function also sets the UART/USART in IRDA mode (IREN bit).</li> <li>• Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx</li> </ul> |

instance.

- Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL\_USART\_DisableLIN() functionClear CLKEN in CR2 using LL\_USART\_DisableSCLKOutput() functionClear SCEN in CR3 using LL\_USART\_DisableSmartcard() functionClear HDSEL in CR3 using LL\_USART\_DisableHalfDuplex() functionConfigure STOP in CR2 using LL\_USART\_SetStopBitsLength() functionSet IREN in CR3 using LL\_USART\_EnableIrda() function
- Other remaining configurations items related to Irda Mode (as Baud Rate, Word length, Power mode, ...) should be set using dedicated functions

Reference Manual to  
LL API cross  
reference:

- CR2 LINEN LL\_USART\_ConfigIrdaMode
- CR2 CLKEN LL\_USART\_ConfigIrdaMode
- CR2 STOP LL\_USART\_ConfigIrdaMode
- CR3 SCEN LL\_USART\_ConfigIrdaMode
- CR3 HDSEL LL\_USART\_ConfigIrdaMode
- CR3 IREN LL\_USART\_ConfigIrdaMode

### **LL\_USART\_ConfigMultiProcessMode**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_ConfigMultiProcessMode(<br/>USART_TypeDef * USARTx)</code></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Function description                              | Perform basic configuration of USART for enabling use in Multi processor Mode (several USARTs connected in a network, one of the USARTs can be the master, its TX output connected to the RX inputs of the other slaves USARTs).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Notes                                             | <ul style="list-style-type: none"> <li>• In MultiProcessor mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,CLKEN bit in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register.</li> <li>• Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear CLKEN in CR2 using LL_USART_DisableSCLKOutput() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() function</li> <li>• Other remaining configurations items related to Multi processor Mode (as Baud Rate, Wake Up Method, Node address, ...) should be set using dedicated functions</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LINEN LL_USART_ConfigMultiProcessMode</li> <li>• CR2 CLKEN LL_USART_ConfigMultiProcessMode</li> <li>• CR3 SCEN LL_USART_ConfigMultiProcessMode</li> <li>• CR3 HDSEL LL_USART_ConfigMultiProcessMode</li> <li>• CR3 IREN LL_USART_ConfigMultiProcessMode</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

**LL\_USART\_IsActiveFlag\_PE**

|                                                   |                                                                                             |
|---------------------------------------------------|---------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_PE<br/>(USART_TypeDef * USARTx)</code> |
| Function description                              | Check if the USART Parity Error Flag is set or not.                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR PE LL_USART_IsActiveFlag_PE</li> </ul>          |

**LL\_USART\_IsActiveFlag\_FE**

|                                                   |                                                                                             |
|---------------------------------------------------|---------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_FE<br/>(USART_TypeDef * USARTx)</code> |
| Function description                              | Check if the USART Framing Error Flag is set or not.                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR FE LL_USART_IsActiveFlag_FE</li> </ul>          |

**LL\_USART\_IsActiveFlag\_NE**

|                                                   |                                                                                             |
|---------------------------------------------------|---------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_NE<br/>(USART_TypeDef * USARTx)</code> |
| Function description                              | Check if the USART Noise error detected Flag is set or not.                                 |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR NF LL_USART_IsActiveFlag_NE</li> </ul>          |

**LL\_USART\_IsActiveFlag\_ORE**

|                                                   |                                                                                              |
|---------------------------------------------------|----------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_ORE<br/>(USART_TypeDef * USARTx)</code> |
| Function description                              | Check if the USART OverRun Error Flag is set or not.                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>           |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR ORE LL_USART_IsActiveFlag_ORE</li> </ul>         |

**LL\_USART\_IsActiveFlag\_IDLE**

Function name **`_STATIC_INLINE uint32_t LL_USART_IsActiveFlag_IDLE(USART_TypeDef * USARTx)`**

Function description Check if the USART IDLE line detected Flag is set or not.

Parameters • **USARTx:** USART Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:  
• SR IDLE LL\_USART\_IsActiveFlag\_IDLE

**LL\_USART\_IsActiveFlag\_RXNE**

Function name **`_STATIC_INLINE uint32_t LL_USART_IsActiveFlag_RXNE(USART_TypeDef * USARTx)`**

Function description Check if the USART Read Data Register Not Empty Flag is set or not.

Parameters • **USARTx:** USART Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:  
• SR RXNE LL\_USART\_IsActiveFlag\_RXNE

**LL\_USART\_IsActiveFlag\_TC**

Function name **`_STATIC_INLINE uint32_t LL_USART_IsActiveFlag_TC(USART_TypeDef * USARTx)`**

Function description Check if the USART Transmission Complete Flag is set or not.

Parameters • **USARTx:** USART Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:  
• SR TC LL\_USART\_IsActiveFlag\_TC

**LL\_USART\_IsActiveFlag\_TXE**

Function name **`_STATIC_INLINE uint32_t LL_USART_IsActiveFlag_TXE(USART_TypeDef * USARTx)`**

Function description Check if the USART Transmit Data Register Empty Flag is set or not.

Parameters • **USARTx:** USART Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:  
• SR TXE LL\_USART\_IsActiveFlag\_TXE

**LL\_USART\_IsActiveFlag\_LBD**

|                                                   |                                                                                                                                                                             |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_LBD(<br/>USART_TypeDef * USARTx)</code></b>                                                                         |
| Function description                              | Check if the USART LIN Break Detection Flag is set or not.                                                                                                                  |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                          |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR LBD LL_USART_IsActiveFlag_LBD</li> </ul>                                                                                        |

**LL\_USART\_IsActiveFlag\_nCTS**

|                                                   |                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_nCTS(<br/>USART_TypeDef * USARTx)</code></b>                                                                                             |
| Function description                              | Check if the USART CTS Flag is set or not.                                                                                                                                                       |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                                               |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• SR CTS LL_USART_IsActiveFlag_nCTS</li> </ul>                                                                                                            |

**LL\_USART\_IsActiveFlag\_SBK**

|                                                   |                                                                                                     |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_SBK(<br/>USART_TypeDef * USARTx)</code></b> |
| Function description                              | Check if the USART Send Break Flag is set or not.                                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                   |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                  |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 SBK LL_USART_IsActiveFlag_SBK</li> </ul>               |

**LL\_USART\_IsActiveFlag\_RWU**

|                      |                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_RWU(<br/>USART_TypeDef * USARTx)</code></b> |
| Function description | Check if the USART Receive Wake Up from mute mode Flag is set or not.                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                   |

---

|                                                   |                                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------|
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR1 RWU LL_USART_IsActiveFlag_RWU</li> </ul> |

### LL\_USART\_ClearFlag\_PE

|                                                   |                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_ClearFlag_PE(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                                                                                         |
| Function description                              | Clear Parity Error Flag.                                                                                                                                                                                                                                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                  |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                                                                                                                                                   |
| Notes                                             | <ul style="list-style-type: none"> <li>Clearing this flag is done by a read access to the USARTx_SR register followed by a read access to the USARTx_DR register.</li> <li>Please also consider that when clearing this flag, other flags as NE, FE, ORE, IDLE would also be cleared.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>SR PE LL_USART_ClearFlag_PE</li> </ul>                                                                                                                                                                                                                    |

### LL\_USART\_ClearFlag\_FE

|                                                   |                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_ClearFlag_FE(<br/>    USART_TypeDef * USARTx)</code>                                                                                                                                                                                                         |
| Function description                              | Clear Framing Error Flag.                                                                                                                                                                                                                                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                  |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                                                                                                                                                   |
| Notes                                             | <ul style="list-style-type: none"> <li>Clearing this flag is done by a read access to the USARTx_SR register followed by a read access to the USARTx_DR register.</li> <li>Please also consider that when clearing this flag, other flags as PE, NE, ORE, IDLE would also be cleared.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>SR FE LL_USART_ClearFlag_FE</li> </ul>                                                                                                                                                                                                                    |

### LL\_USART\_ClearFlag\_NE

|                      |                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_USART_ClearFlag_NE(<br/>    USART_TypeDef * USARTx)</code>                                                                 |
| Function description | Clear Noise detected Flag.                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                          |
| Return values        | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>Clearing this flag is done by a read access to the USARTx_SR register followed by a read access to the</li> </ul> |

Reference Manual to  
LL API cross  
reference:

- USARTx\_DR register.
- Please also consider that when clearing this flag, other flags as PE, FE, ORE, IDLE would also be cleared.

- SR NF LL\_USART\_ClearFlag\_NE

### **LL\_USART\_ClearFlag\_ORE**

Function name

**`_STATIC_INLINE void LL_USART_ClearFlag_ORE  
(USART_TypeDef * USARTx)`**

Function description

Clear OverRun Error Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the USARTx\_SR register followed by a read access to the USARTx\_DR register.
- Please also consider that when clearing this flag, other flags as PE, NE, FE, IDLE would also be cleared.

Reference Manual to  
LL API cross  
reference:

- SR ORE LL\_USART\_ClearFlag\_ORE

### **LL\_USART\_ClearFlag\_IDLE**

Function name

**`_STATIC_INLINE void LL_USART_ClearFlag_IDLE  
(USART_TypeDef * USARTx)`**

Function description

Clear IDLE line detected Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the USARTx\_SR register followed by a read access to the USARTx\_DR register.
- Please also consider that when clearing this flag, other flags as PE, NE, FE, ORE would also be cleared.

Reference Manual to  
LL API cross  
reference:

- SR IDLE LL\_USART\_ClearFlag\_IDLE

### **LL\_USART\_ClearFlag\_TC**

Function name

**`_STATIC_INLINE void LL_USART_ClearFlag_TC  
(USART_TypeDef * USARTx)`**

Function description

Clear Transmission Complete Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

- Reference Manual to  
LL API cross  
reference:
- SR TC LL\_USART\_ClearFlag\_TC

### **LL\_USART\_ClearFlag\_RXNE**

Function name      **`_STATIC_INLINE void LL_USART_ClearFlag_RXNE  
(USART_TypeDef * USARTx)`**

Function description      Clear RX Not Empty Flag.

Parameters      • **USARTx:** USART Instance

Return values      • **None:**

- Reference Manual to  
LL API cross  
reference:
- SR RXNE LL\_USART\_ClearFlag\_RXNE

### **LL\_USART\_ClearFlag\_LBD**

Function name      **`_STATIC_INLINE void LL_USART_ClearFlag_LBD  
(USART_TypeDef * USARTx)`**

Function description      Clear LIN Break Detection Flag.

Parameters      • **USARTx:** USART Instance

Return values      • **None:**

Notes      • Macro IS\_UART\_LIN\_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

- Reference Manual to  
LL API cross  
reference:
- SR LBD LL\_USART\_ClearFlag\_LBD

### **LL\_USART\_ClearFlag\_nCTS**

Function name      **`_STATIC_INLINE void LL_USART_ClearFlag_nCTS  
(USART_TypeDef * USARTx)`**

Function description      Clear CTS Interrupt Flag.

Parameters      • **USARTx:** USART Instance

Return values      • **None:**

Notes      • Macro IS\_UART\_HWFLOW\_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

- Reference Manual to  
LL API cross  
reference:
- SR CTS LL\_USART\_ClearFlag\_nCTS

### **LL\_USART\_EnableIT\_IDLE**

Function name      **`_STATIC_INLINE void LL_USART_EnableIT_IDLE  
(USART_TypeDef * USARTx)`**

---

|                                                   |                                                                                       |
|---------------------------------------------------|---------------------------------------------------------------------------------------|
| Function description                              | Enable IDLE Interrupt.                                                                |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>    |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 IDLEIE LL_USART_EnableIT_IDLE</li> </ul> |

### LL\_USART\_EnableIT\_RXNE

|                                                   |                                                                                       |
|---------------------------------------------------|---------------------------------------------------------------------------------------|
| Function name                                     | <b>__STATIC_INLINE void LL_USART_EnableIT_RXNE<br/>(USART_TypeDef * USARTx)</b>       |
| Function description                              | Enable RX Not Empty Interrupt.                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>    |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                      |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 RXNEIE LL_USART_EnableIT_RXNE</li> </ul> |

### LL\_USART\_EnableIT\_TC

|                                                   |                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------|
| Function name                                     | <b>__STATIC_INLINE void LL_USART_EnableIT_TC<br/>(USART_TypeDef * USARTx)</b>      |
| Function description                              | Enable Transmission Complete Interrupt.                                            |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 TCIE LL_USART_EnableIT_TC</li> </ul>  |

### LL\_USART\_EnableIT\_TXE

|                                                   |                                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------|
| Function name                                     | <b>__STATIC_INLINE void LL_USART_EnableIT_TXE<br/>(USART_TypeDef * USARTx)</b>      |
| Function description                              | Enable TX Empty Interrupt.                                                          |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>  |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                    |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 TXEIE LL_USART_EnableIT_TXE</li> </ul> |

### LL\_USART\_EnableIT\_PE

|                      |                                                                               |
|----------------------|-------------------------------------------------------------------------------|
| Function name        | <b>__STATIC_INLINE void LL_USART_EnableIT_PE<br/>(USART_TypeDef * USARTx)</b> |
| Function description | Enable Parity Error Interrupt.                                                |

|                                                   |                                                                                 |
|---------------------------------------------------|---------------------------------------------------------------------------------|
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                  |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR1 PEIE LL_USART_EnableIT_PE</li> </ul> |

### LL\_USART\_EnableIT\_LBD

|                                                   |                                                                                                                                                                           |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_EnableIT_LBD<br/>(USART_TypeDef * USARTx)</code></b>                                                                                |
| Function description                              | Enable LIN Break Detection Interrupt.                                                                                                                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                           |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                            |
| Notes                                             | <ul style="list-style-type: none"> <li>Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR2 LBDIE LL_USART_EnableIT_LBD</li> </ul>                                                                                         |

### LL\_USART\_EnableIT\_ERROR

|                                                   |                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_USART_EnableIT_ERROR<br/>(USART_TypeDef * USARTx)</code></b>                                                                                                                                                                                                                                                    |
| Function description                              | Enable Error Interrupt.                                                                                                                                                                                                                                                                                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                                                                                                                                                                                                 |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                  |
| Notes                                             | <ul style="list-style-type: none"> <li>When set, Error Interrupt Enable Bit is enabling interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the USARTx_SR register). 0: Interrupt is inhibited 1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the USARTx_SR register.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR3 EIE LL_USART_EnableIT_ERROR</li> </ul>                                                                                                                                                                                                                                                               |

### LL\_USART\_EnableIT\_CTS

|                      |                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE void LL_USART_EnableIT_CTS<br/>(USART_TypeDef * USARTx)</code></b>                                                                |
| Function description | Enable CTS Interrupt.                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                           |
| Return values        | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature</li> </ul> |

is supported by the USARTx instance.

Reference Manual to  
LL API cross  
reference:

- CR3 CTSIE LL\_USART\_EnableIT\_CTS

### **LL\_USART\_DisableIT\_IDLE**

Function name

**`_STATIC_INLINE void LL_USART_DisableIT_IDLE  
(USART_TypeDef * USARTx)`**

Function description

Disable IDLE Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to

LL API cross  
reference:

- CR1 IDLEIE LL\_USART\_DisableIT\_IDLE

### **LL\_USART\_DisableIT\_RXNE**

Function name

**`_STATIC_INLINE void LL_USART_DisableIT_RXNE  
(USART_TypeDef * USARTx)`**

Function description

Disable RX Not Empty Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to

LL API cross  
reference:

- CR1 RXNEIE LL\_USART\_DisableIT\_RXNE

### **LL\_USART\_DisableIT\_TC**

Function name

**`_STATIC_INLINE void LL_USART_DisableIT_TC  
(USART_TypeDef * USARTx)`**

Function description

Disable Transmission Complete Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to

LL API cross  
reference:

- CR1 TCIE LL\_USART\_DisableIT\_TC

### **LL\_USART\_DisableIT\_TXE**

Function name

**`_STATIC_INLINE void LL_USART_DisableIT_TXE  
(USART_TypeDef * USARTx)`**

Function description

Disable TX Empty Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

- Reference Manual to  
LL API cross  
reference:
- CR1 TXEIE LL\_USART\_DisableIT\_TXE

### **LL\_USART\_DisableIT\_PE**

Function name      **\_\_STATIC\_INLINE void LL\_USART\_DisableIT\_PE  
(USART\_TypeDef \* USARTx)**

Function description      Disable Parity Error Interrupt.

- Parameters      • **USARTx:** USART Instance

- Return values      • **None:**

- Reference Manual to  
LL API cross  
reference:
- CR1 PEIE LL\_USART\_DisableIT\_PE

### **LL\_USART\_DisableIT\_LBD**

Function name      **\_\_STATIC\_INLINE void LL\_USART\_DisableIT\_LBD  
(USART\_TypeDef \* USARTx)**

Function description      Disable LIN Break Detection Interrupt.

- Parameters      • **USARTx:** USART Instance

- Return values      • **None:**

- Notes      • Macro IS\_UART\_LIN\_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

- Reference Manual to  
LL API cross  
reference:
- CR2 LBDIE LL\_USART\_DisableIT\_LBD

### **LL\_USART\_DisableIT\_ERROR**

Function name      **\_\_STATIC\_INLINE void LL\_USART\_DisableIT\_ERROR  
(USART\_TypeDef \* USARTx)**

Function description      Disable Error Interrupt.

- Parameters      • **USARTx:** USART Instance

- Return values      • **None:**

- Notes      • When set, Error Interrupt Enable Bit is enabling interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the USARTx\_SR register).  
0: Interrupt is inhibited 1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the USARTx\_SR register.

- Reference Manual to  
LL API cross  
reference:
- CR3 EIE LL\_USART\_DisableIT\_ERROR

**LL\_USART\_DisableIT\_CTS**

|                                                   |                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_DisableIT_CTS<br/>(USART_TypeDef * USARTx)</code>                                                                                                            |
| Function description                              | Disable CTS Interrupt.                                                                                                                                                                           |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                                                |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                 |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR3 CTSIE LL_USART_DisableIT_CTS</li> </ul>                                                                                                             |

**LL\_USART\_IsEnabledIT\_IDLE**

|                                                   |                                                                                              |
|---------------------------------------------------|----------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_IDLE<br/>(USART_TypeDef * USARTx)</code> |
| Function description                              | Check if the USART IDLE Interrupt source is enabled or disabled.                             |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>           |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 IDLEIE LL_USART_IsEnabledIT_IDLE</li> </ul>     |

**LL\_USART\_IsEnabledIT\_RXNE**

|                                                   |                                                                                              |
|---------------------------------------------------|----------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_RXNE<br/>(USART_TypeDef * USARTx)</code> |
| Function description                              | Check if the USART RX Not Empty Interrupt is enabled or disabled.                            |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>           |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 RXNEIE LL_USART_IsEnabledIT_RXNE</li> </ul>     |

**LL\_USART\_IsEnabledIT\_TC**

|                      |                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_TC<br/>(USART_TypeDef * USARTx)</code> |
| Function description | Check if the USART Transmission Complete Interrupt is enabled or disabled.                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>          |
| Return values        | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>         |

- Reference Manual to  
LL API cross  
reference:
- CR1 TCIE LL\_USART\_IsEnabledIT\_TC

### **LL\_USART\_IsEnabledIT\_TXE**

|                                                   |                                                                                                 |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_TXE(<br/>    USART_TypeDef * USARTx)</code> |
| Function description                              | Check if the USART TX Empty Interrupt is enabled or disabled.                                   |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>               |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>              |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 TXEIE LL_USART_IsEnabledIT_TXE</li> </ul>          |

### **LL\_USART\_IsEnabledIT\_PE**

|                                                   |                                                                                                |
|---------------------------------------------------|------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_PE(<br/>    USART_TypeDef * USARTx)</code> |
| Function description                              | Check if the USART Parity Error Interrupt is enabled or disabled.                              |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>              |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>             |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 PEIE LL_USART_IsEnabledIT_PE</li> </ul>           |

### **LL\_USART\_IsEnabledIT\_LBD**

|                                                   |                                                                                                                                                                             |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_LBD(<br/>    USART_TypeDef * USARTx)</code>                                                                             |
| Function description                              | Check if the USART LIN Break Detection Interrupt is enabled or disabled.                                                                                                    |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTx:</b> USART Instance</li> </ul>                                                                                           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                                                                                          |
| Notes                                             | <ul style="list-style-type: none"> <li>• Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR2 LBDIE LL_USART_IsEnabledIT_LBD</li> </ul>                                                                                      |

### **LL\_USART\_IsEnabledIT\_ERROR**

|                      |                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_ERROR(<br/>    USART_TypeDef * USARTx)</code> |
| Function description | Check if the USART Error Interrupt is enabled or disabled.                                        |

---

|                                                   |                                                                                      |
|---------------------------------------------------|--------------------------------------------------------------------------------------|
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>      |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>     |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR3 EIE LL_USART_IsEnabledIT_ERROR</li> </ul> |

### LL\_USART\_IsEnabledIT\_CTS

|                                                   |                                                                                                                                                                                                |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>_STATIC_INLINE uint32_t LL_USART_IsEnabledIT_CTS<br/>(USART_TypeDef * USARTx)</code>                                                                                                     |
| Function description                              | Check if the USART CTS Interrupt is enabled or disabled.                                                                                                                                       |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                                                                                                |
| Return values                                     | <ul style="list-style-type: none"> <li><b>State:</b> of bit (1 or 0).</li> </ul>                                                                                                               |
| Notes                                             | <ul style="list-style-type: none"> <li>Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR3 CTSIE LL_USART_IsEnabledIT_CTS</li> </ul>                                                                                                           |

### LL\_USART\_EnableDMAReq\_RX

|                                                   |                                                                                        |
|---------------------------------------------------|----------------------------------------------------------------------------------------|
| Function name                                     | <code>_STATIC_INLINE void LL_USART_EnableDMAReq_RX<br/>(USART_TypeDef * USARTx)</code> |
| Function description                              | Enable DMA Mode for reception.                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>        |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                         |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR3 DMAR LL_USART_EnableDMAReq_RX</li> </ul>    |

### LL\_USART\_DisableDMAReq\_RX

|                                                   |                                                                                         |
|---------------------------------------------------|-----------------------------------------------------------------------------------------|
| Function name                                     | <code>_STATIC_INLINE void LL_USART_DisableDMAReq_RX<br/>(USART_TypeDef * USARTx)</code> |
| Function description                              | Disable DMA Mode for reception.                                                         |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>         |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                          |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>CR3 DMAR LL_USART_DisableDMAReq_RX</li> </ul>    |

### LL\_USART\_IsEnabledDMAReq\_RX

|               |                                                                                               |
|---------------|-----------------------------------------------------------------------------------------------|
| Function name | <code>_STATIC_INLINE uint32_t LL_USART_IsEnabledDMAReq_RX<br/>(USART_TypeDef * USARTx)</code> |
|---------------|-----------------------------------------------------------------------------------------------|

|                                                   |                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------|
| Function description                              | Check if DMA Mode is enabled for reception.                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | CR3 DMAR LL_USART_IsEnabledDMAReq_RX                                               |

### LL\_USART\_EnableDMAReq\_TX

|                                                   |                                                                                             |
|---------------------------------------------------|---------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_EnableDMAReq_TX(<br/>    USART_TypeDef * USARTx)</code> |
| Function description                              | Enable DMA Mode for transmission.                                                           |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>          |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                            |
| Reference Manual to<br>LL API cross<br>reference: | CR3 DMAT LL_USART_EnableDMAReq_TX                                                           |

### LL\_USART\_DisableDMAReq\_TX

|                                                   |                                                                                              |
|---------------------------------------------------|----------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_DisableDMAReq_TX(<br/>    USART_TypeDef * USARTx)</code> |
| Function description                              | Disable DMA Mode for transmission.                                                           |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>           |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                             |
| Reference Manual to<br>LL API cross<br>reference: | CR3 DMAT LL_USART_DisableDMAReq_TX                                                           |

### LL\_USART\_IsEnabledDMAReq\_TX

|                                                   |                                                                                                    |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_USART_IsEnabledDMAReq_TX(<br/>    USART_TypeDef * USARTx)</code> |
| Function description                              | Check if DMA Mode is enabled for transmission.                                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                 |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                 |
| Reference Manual to<br>LL API cross<br>reference: | CR3 DMAT LL_USART_IsEnabledDMAReq_TX                                                               |

### LL\_USART\_DMA\_GetRegAddr

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_USART_DMA_GetRegAddr(<br/>    USART_TypeDef * USARTx)</code> |
| Function description | Get the data register address used for DMA transfer.                                           |

|                                                   |                                                                                                                              |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                                              |
| Return values                                     | <ul style="list-style-type: none"> <li><b>Address:</b> of data register</li> </ul>                                           |
| Notes                                             | <ul style="list-style-type: none"> <li>Address of Data Register is valid for both Transmit and Receive transfers.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DR DR LL_USART_DMA_GetRegAddr</li> </ul>                                              |

### LL\_USART\_ReceiveData8

|                                                   |                                                                                                         |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint8_t LL_USART_ReceiveData8(<br/>    USART_TypeDef * USARTx)</code>             |
| Function description                              | Read Receiver Data register (Receive Data value, 8 bits)                                                |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                         |
| Return values                                     | <ul style="list-style-type: none"> <li><b>Value:</b> between Min_Data=0x00 and Max_Data=0xFF</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DR DR LL_USART_ReceiveData8</li> </ul>                           |

### LL\_USART\_ReceiveData9

|                                                   |                                                                                                          |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint16_t LL_USART_ReceiveData9(<br/>    USART_TypeDef * USARTx)</code>             |
| Function description                              | Read Receiver Data register (Receive Data value, 9 bits)                                                 |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> </ul>                          |
| Return values                                     | <ul style="list-style-type: none"> <li><b>Value:</b> between Min_Data=0x00 and Max_Data=0x1FF</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DR DR LL_USART_ReceiveData9</li> </ul>                            |

### LL\_USART\_TransmitData8

|                                                   |                                                                                                                                                |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_TransmitData8(<br/>    USART_TypeDef * USARTx, uint8_t Value)</code>                                       |
| Function description                              | Write in Transmitter Data Register (Transmit Data value, 8 bits)                                                                               |
| Parameters                                        | <ul style="list-style-type: none"> <li><b>USARTx:</b> USART Instance</li> <li><b>Value:</b> between Min_Data=0x00 and Max_Data=0xFF</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                 |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>DR DR LL_USART_TransmitData8</li> </ul>                                                                 |

### LL\_USART\_TransmitData9

|               |                                                                                                           |
|---------------|-----------------------------------------------------------------------------------------------------------|
| Function name | <code>__STATIC_INLINE void LL_USART_TransmitData9(<br/>    USART_TypeDef * USARTx, uint16_t Value)</code> |
|---------------|-----------------------------------------------------------------------------------------------------------|

|                                                   |                                                                                                                                                      |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function description                              | Write in Transmitter Data Register (Transmit Data value, 9 bits)                                                                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> <li>• <b>Value:</b> between Min_Data=0x00 and Max_Data=0x1FF</li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                     |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• DR DR LL_USART_TransmitData9</li> </ul>                                                                     |

### LL\_USART\_RequestBreakSending

|                                                   |                                                                                                               |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_RequestBreakSending(<br/>                  USART_TypeDef * USARTx)</code> |
| Function description                              | Request Break sending.                                                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                              |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 SBK LL_USART_RequestBreakSending</li> </ul>                      |

### LL\_USART\_RequestEnterMuteMode

|                                                   |                                                                                                                |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_RequestEnterMuteMode(<br/>                  USART_TypeDef * USARTx)</code> |
| Function description                              | Put USART in Mute mode.                                                                                        |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                             |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                               |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 RWU LL_USART_RequestEnterMuteMode</li> </ul>                      |

### LL\_USART\_RequestExitMuteMode

|                                                   |                                                                                                               |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_USART_RequestExitMuteMode(<br/>                  USART_TypeDef * USARTx)</code> |
| Function description                              | Put USART in Active mode.                                                                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                              |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR1 RWU LL_USART_RequestExitMuteMode</li> </ul>                      |

### LL\_USART\_DeInit

|                      |                                                                    |
|----------------------|--------------------------------------------------------------------|
| Function name        | <code>ErrorStatus LL_USART_DeInit(USART_TypeDef * USARTx)</code>   |
| Function description | De-initialize USART registers (Registers restored to their default |

values).

- |               |                                                                                                                                                                                                                                                                   |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> </ul>                                                                                                                                                                                |
| Return values | <ul style="list-style-type: none"> <li>• <b>An:</b> ErrorStatus enumeration value:           <ul style="list-style-type: none"> <li>– SUCCESS: USART registers are de-initialized</li> <li>– ERROR: USART registers are not de-initialized</li> </ul> </li> </ul> |

### **LL\_USART\_Init**

|                      |                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>ErrorStatus LL_USART_Init (USART_TypeDef * USARTTx,<br/>LL_USART_InitTypeDef * USART_InitStruct)</b>                                                                                                                                                                                                                                                                                       |
| Function description | Initialize USART registers according to the specified parameters in USART_InitStruct.                                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> <li>• <b>USART_InitStruct:</b> pointer to a LL_USART_InitTypeDef structure that contains the configuration information for the specified USART peripheral.</li> </ul>                                                                                                                                            |
| Return values        | <ul style="list-style-type: none"> <li>• <b>An:</b> ErrorStatus enumeration value:           <ul style="list-style-type: none"> <li>– SUCCESS: USART registers are initialized according to USART_InitStruct content</li> <li>– ERROR: Problem occurred during USART Registers initialization</li> </ul> </li> </ul>                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• As some bits in USART configuration registers can only be written when the USART is disabled (USART_CR1_UE bit =0), USART IP should be in disabled state prior calling this function. Otherwise, ERROR result will be returned.</li> <li>• Baud rate value stored in USART_InitStruct BaudRate field, should be valid (different from 0).</li> </ul> |

### **LL\_USART\_StructInit**

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>void LL_USART_StructInit (LL_USART_InitTypeDef *<br/>USART_InitStruct)</b>                                                                                        |
| Function description | Set each LL_USART_InitTypeDef field to default value.                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USART_InitStruct:</b> pointer to a LL_USART_InitTypeDef structure whose fields will be set to default values.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                     |

### **LL\_USART\_ClockInit**

|                      |                                                                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>ErrorStatus LL_USART_ClockInit (USART_TypeDef * USARTTx,<br/>LL_USART_ClockInitTypeDef * USART_ClockInitStruct)</b>                                                                                                                                             |
| Function description | Initialize USART Clock related settings according to the specified parameters in the USART_ClockInitStruct.                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>USARTTx:</b> USART Instance</li> <li>• <b>USART_ClockInitStruct:</b> pointer to a LL_USART_ClockInitTypeDef structure that contains the Clock configuration information for the specified USART peripheral.</li> </ul> |

|               |                                                                                                                                                                                                                                                                                                                                               |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"> <li><b>An:</b> ErrorStatus enumeration value:           <ul style="list-style-type: none"> <li>SUCCESS: USART registers related to Clock settings are initialized according to USART_ClockInitStruct content</li> <li>ERROR: Problem occurred during USART Registers initialization</li> </ul> </li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>As some bits in USART configuration registers can only be written when the USART is disabled (USART_CR1_UE bit =0), USART IP should be in disabled state prior calling this function. Otherwise, ERROR result will be returned.</li> </ul>                                                             |

### LL\_USART\_ClockStructInit

|                      |                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>void LL_USART_ClockStructInit<br/>(LL_USART_ClockInitTypeDef * USART_ClockInitStruct)</code>                                                                           |
| Function description | Set each field of a LL_USART_ClockInitTypeDef type structure to default value.                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li><b>USART_ClockInitStruct:</b> pointer to a LL_USART_ClockInitTypeDef structure whose fields will be set to default values.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>None:</b></li> </ul>                                                                                                               |

## 67.3 USART Firmware driver defines

### 67.3.1 USART

#### *Clock Signal*

`LL_USART_CLOCK_DISABLE` Clock signal not provided

`LL_USART_CLOCK_ENABLE` Clock signal provided

#### *Datawidth*

`LL_USART_DATAWIDTH_8B` 8 bits word length : Start bit, 8 data bits, n stop bits

`LL_USART_DATAWIDTH_9B` 9 bits word length : Start bit, 9 data bits, n stop bits

#### *Communication Direction*

`LL_USART_DIRECTION_NONE` Transmitter and Receiver are disabled

`LL_USART_DIRECTION_RX` Transmitter is disabled and Receiver is enabled

`LL_USART_DIRECTION_TX` Transmitter is enabled and Receiver is disabled

`LL_USART_DIRECTION_TX_RX` Transmitter and Receiver are enabled

#### *Get Flags Defines*

`LL_USART_SR_PE` Parity error flag

`LL_USART_SR_FE` Framing error flag

`LL_USART_SR_NE` Noise detected flag

`LL_USART_SR_ORE` Overrun error flag

`LL_USART_SR_IDLE` Idle line detected flag

---

|                  |                                   |
|------------------|-----------------------------------|
| LL_USART_SR_RXNE | Read data register not empty flag |
| LL_USART_SR_TC   | Transmission complete flag        |
| LL_USART_SR_TXE  | Transmit data register empty flag |
| LL_USART_SR_LBD  | LIN break detection flag          |
| LL_USART_SR_CTS  | CTS flag                          |

#### **Hardware Control**

|                            |                                                                                        |
|----------------------------|----------------------------------------------------------------------------------------|
| LL_USART_HWCONTROL_NONE    | CTS and RTS hardware flow control disabled                                             |
| LL_USART_HWCONTROL_RTS     | RTS output enabled, data is only requested when there is space in the receive buffer   |
| LL_USART_HWCONTROL_CTS     | CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0) |
| LL_USART_HWCONTROL_RTS_CTS | CTS and RTS hardware flow control enabled                                              |

#### **IrDA Power**

|                            |                        |
|----------------------------|------------------------|
| LL_USART_IRDA_POWER_NORMAL | IrDA normal power mode |
| LL_USART_IRDA_POWER_LOW    | IrDA low power mode    |

#### **IT Defines**

|                     |                                               |
|---------------------|-----------------------------------------------|
| LL_USART_CR1_IDLEIE | IDLE interrupt enable                         |
| LL_USART_CR1_RXNEIE | Read data register not empty interrupt enable |
| LL_USART_CR1_TCIE   | Transmission complete interrupt enable        |
| LL_USART_CR1_TXEIE  | Transmit data register empty interrupt enable |
| LL_USART_CR1_PEIE   | Parity error                                  |
| LL_USART_CR2_LBDIE  | LIN break detection interrupt enable          |
| LL_USART_CR3_EIE    | Error interrupt enable                        |
| LL_USART_CR3_CTSIE  | CTS interrupt enable                          |

#### **Last Clock Pulse**

|                                 |                                                                    |
|---------------------------------|--------------------------------------------------------------------|
| LL_USART_LASTCLKPULSE_NO_OUTPUT | The clock pulse of the last data bit is not output to the SCLK pin |
| LL_USART_LASTCLKPULSE_OUTPUT    | The clock pulse of the last data bit is output to the SCLK pin     |

#### **LIN Break Detection Length**

|                              |                                        |
|------------------------------|----------------------------------------|
| LL_USART_LINBREAK_DETECT_10B | 10-bit break detection method selected |
| LL_USART_LINBREAK_DETECT_11B | 11-bit break detection method selected |

#### **Oversampling**

|                          |                    |
|--------------------------|--------------------|
| LL_USART_OVERSAMPLING_16 | Oversampling by 16 |
| LL_USART_OVERSAMPLING_8  | Oversampling by 8  |

#### **Parity Control**

|                      |                                                    |
|----------------------|----------------------------------------------------|
| LL_USART_PARITY_NONE | Parity control disabled                            |
| LL_USART_PARITY EVEN | Parity control enabled and Even Parity is selected |

`LL_USART_PARITY_ODD` Parity control enabled and Odd Parity is selected

#### **Clock Phase**

`LL_USART_PHASE_1EDGE` The first clock transition is the first data capture edge

`LL_USART_PHASE_2EDGE` The second clock transition is the first data capture edge

#### **Clock Polarity**

`LL_USART_POLARITY_LOW` Steady low value on SCLK pin outside transmission window

`LL_USART_POLARITY_HIGH` Steady high value on SCLK pin outside transmission window

#### **Stop Bits**

`LL_USART_STOPBITS_0_5` 0.5 stop bit

`LL_USART_STOPBITS_1` 1 stop bit

`LL_USART_STOPBITS_1_5` 1.5 stop bits

`LL_USART_STOPBITS_2` 2 stop bits

#### **Wakeup**

`LL_USART_WAKEUP_IDLELINE` USART wake up from Mute mode on Idle Line

`LL_USART_WAKEUP_ADDRESSMARK` USART wake up from Mute mode on Address Mark

#### **Exported Macros Helper**

`_LL_USART_DIV_SAMPLING8_100`

##### **Description:**

- Compute USARTDIV value according to Peripheral Clock and expected Baud Rate in 8 bits sampling mode (32 bits value of USARTDIV is returned)

##### **Parameters:**

- `_PERIPHCLK_`: Peripheral Clock frequency used for USART instance
- `_BAUDRATE_`: Baud rate value to achieve

##### **Return value:**

- USARTDIV: value to be used for BRR register filling in OverSampling\_8 case

`_LL_USART_DIVMANT_SAMPLING8`

`_LL_USART_DIVFRAQ_SAMPLING8`

`_LL_USART_DIV_SAMPLING8`

`_LL_USART_DIV_SAMPLING16_100`

##### **Description:**

- Compute USARTDIV value according to Peripheral Clock and expected Baud Rate in 16 bits sampling mode (32 bits value of USARTDIV is returned)

##### **Parameters:**

- PERIPHCLK: Peripheral Clock frequency used for USART instance
- BAUDRATE: Baud rate value to achieve

**Return value:**

- USARTDIV: value to be used for BRR register filling in OverSampling\_16 case

\_LL\_USART\_DIVMANT\_SAMPLING16

\_LL\_USART\_DIVFRAQ\_SAMPLING16

\_LL\_USART\_DIV\_SAMPLING16

**Common Write and read registers Macros**

**LL\_USART\_WriteReg Description:**

- Write a value in USART register.

**Parameters:**

- INSTANCE: USART Instance
- REG: Register to be written
- VALUE: Value to be written in the register

**Return value:**

- None

**LL\_USART\_ReadReg Description:**

- Read a value in USART register.

**Parameters:**

- INSTANCE: USART Instance
- REG: Register to be read

**Return value:**

- Register: value

## 68 LL UTILS Generic Driver

### 68.1 UTILS Firmware driver registers structures

#### 68.1.1 LL\_UTILS\_PLLInitTypeDef

##### Data Fields

- *uint32\_t PLLMul*
- *uint32\_t PLLDiv*

##### Field Documentation

- *uint32\_t LL\_UTILS\_PLLInitTypeDef::PLLMul*  
Multiplication factor for PLL VCO input clock. This parameter can be a value of [RCC\\_LL\\_EC\\_PLL\\_MUL](#)This feature can be modified afterwards using unitary function [LL\\_RCC\\_PLL\\_ConfigDomain\\_SYS\(\)](#).
- *uint32\_t LL\_UTILS\_PLLInitTypeDef::PLLDiv*  
Division factor for PLL VCO output clock. This parameter can be a value of [RCC\\_LL\\_EC\\_PLL\\_DIV](#)This feature can be modified afterwards using unitary function [LL\\_RCC\\_PLL\\_ConfigDomain\\_SYS\(\)](#).

#### 68.1.2 LL\_UTILS\_ClkInitTypeDef

##### Data Fields

- *uint32\_t AHBCLKDivider*
- *uint32\_t APB1CLKDivider*
- *uint32\_t APB2CLKDivider*

##### Field Documentation

- *uint32\_t LL\_UTILS\_ClkInitTypeDef::AHBCLKDivider*  
The AHB clock (HCLK) divider. This clock is derived from the system clock (SYSCLK). This parameter can be a value of [RCC\\_LL\\_EC\\_SYSCLK\\_DIV](#)This feature can be modified afterwards using unitary function [LL\\_RCC\\_SetAHBPrescaler\(\)](#).
- *uint32\_t LL\_UTILS\_ClkInitTypeDef::APB1CLKDivider*  
The APB1 clock (PCLK1) divider. This clock is derived from the AHB clock (HCLK). This parameter can be a value of [RCC\\_LL\\_EC\\_APB1\\_DIV](#)This feature can be modified afterwards using unitary function [LL\\_RCC\\_SetAPB1Prescaler\(\)](#).
- *uint32\_t LL\_UTILS\_ClkInitTypeDef::APB2CLKDivider*  
The APB2 clock (PCLK2) divider. This clock is derived from the AHB clock (HCLK). This parameter can be a value of [RCC\\_LL\\_EC\\_APB2\\_DIV](#)This feature can be modified afterwards using unitary function [LL\\_RCC\\_SetAPB2Prescaler\(\)](#).

### 68.2 UTILS Firmware driver API description

#### 68.2.1 System Configuration functions

System, AHB and APB buses clocks configuration

- The maximum frequency of the SYSCLK, HCLK, PCLK1 and PCLK2 is 32000000 Hz.

This section contains the following APIs:

- [LL\\_SetSystemCoreClock\(\)](#)
- [LL\\_PLL\\_ConfigSystemClock\\_HSI\(\)](#)

- [\*\*LL\\_PLL\\_ConfigSystemClock\\_HSE\(\)\*\*](#)

## 68.2.2 Detailed description of functions

### **LL\_GetUID\_Word0**

|                      |                                                                       |
|----------------------|-----------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE uint32_t LL_GetUID_Word0 (void )</code></b>  |
| Function description | Get Word0 of the unique device identifier (UID based on 96 bits)      |
| Return values        | <ul style="list-style-type: none"> <li>• <b>UID[31:0]:</b></li> </ul> |

### **LL\_GetUID\_Word1**

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE uint32_t LL_GetUID_Word1 (void )</code></b>   |
| Function description | Get Word1 of the unique device identifier (UID based on 96 bits)       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>UID[63:32]:</b></li> </ul> |

### **LL\_GetUID\_Word2**

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE uint32_t LL_GetUID_Word2 (void )</code></b>   |
| Function description | Get Word2 of the unique device identifier (UID based on 96 bits)       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>UID[95:64]:</b></li> </ul> |

### **LL\_GetFlashSize**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE uint32_t LL_GetFlashSize (void )</code></b>                                                                                                                                                                                                                                                                                                                                                                                       |
| Function description | Get Flash memory size.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Return values        | <ul style="list-style-type: none"> <li>• <b>FLASH_SIZE[15:0]:</b> Flash memory size</li> </ul>                                                                                                                                                                                                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• For DEV_ID = 0x416 or 0x427 or 0x429 or 0x437, this field value indicates the Flash memory size of the device in Kbytes. Example: 0x0080 = 128 Kbytes. For DEV_ID = 0x436, the field value can be '0' or '1', with '0' for 384 Kbytes and '1' for 256 Kbytes.</li> <li>• For DEV_ID = 0x429, only LSB part of F_SIZE: F_SIZE[7:0] is valid. The MSB part F_SIZE[15:8] is reserved and must be ignored.</li> </ul> |

### **LL\_InitTick**

|                      |                                                                                                                                                                                        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>__STATIC_INLINE void LL_InitTick (uint32_t HCLKFrequency, uint32_t Ticks)</code></b>                                                                                          |
| Function description | This function configures the Cortex-M SysTick source of the time base.                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>HCLKFrequency:</b> HCLK frequency in Hz (can be calculated thanks to RCC helper macro)</li> <li>• <b>Ticks:</b> Number of ticks</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• When a RTOS is used, it is recommended to avoid changing the SysTick configuration by calling this function, for a delay</li> </ul>           |

use rather osDelay RTOS service.

### **LL\_Init1msTick**

|                      |                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>void LL_Init1msTick (uint32_t HCLKFrequency)</b>                                                                                                                                                                                                                                                                         |
| Function description | This function configures the Cortex-M SysTick source to have 1ms time base.                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>HCLKFrequency:</b> HCLK frequency in Hz</li> </ul>                                                                                                                                                                                                                              |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• When a RTOS is used, it is recommended to avoid changing the Systick configuration by calling this function, for a delay use rather osDelay RTOS service.</li> <li>• HCLK frequency can be calculated thanks to RCC helper macro or function LL_RCC_GetSystemClocksFreq</li> </ul> |

### **LL\_mDdelay**

|                      |                                                                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>void LL_mDdelay (uint32_t Delay)</b>                                                                                                                                                                                                                                   |
| Function description | This function provides accurate delay (in milliseconds) based on SysTick counter flag.                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Delay:</b> specifies the delay time length, in milliseconds.</li> </ul>                                                                                                                                                       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• When a RTOS is used, it is recommended to avoid using blocking delay and use rather osDelay service.</li> <li>• To respect 1ms timebase, user should call LL_Init1msTick function which will configure Systick to 1ms</li> </ul> |

### **LL\_SetSystemCoreClock**

|                      |                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>void LL_SetSystemCoreClock (uint32_t HCLKFrequency)</b>                                                                                    |
| Function description | This function sets directly SystemCoreClock CMSIS variable.                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>HCLKFrequency:</b> HCLK frequency in Hz (can be calculated thanks to RCC helper macro)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• Variable can be calculated also through SystemCoreClockUpdate function.</li> </ul>                   |

### **LL\_PLL\_ConfigSystemClock\_HSI**

|                      |                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b>ErrorStatus LL_PLL_ConfigSystemClock_HSI<br/>(LL_UTILS_PLLInitTypeDef * UTILS_PLLInitStruct,<br/>LL_UTILS_ClkInitTypeDef * UTILS_ClkInitStruct)</b>                                  |
| Function description | This function configures system clock with HSI as clock source of the PLL.                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>UTILS_PLLInitStruct:</b> pointer to a LL_UTILS_PLLInitTypeDef structure that contains the configuration information for the PLL.</li> </ul> |

- |                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Return values</p> | <ul style="list-style-type: none"> <li>• <b>UTILS_ClkInitStruct:</b> pointer to a LL_UTILS_ClkInitTypeDef structure that contains the configuration information for the BUS prescalers.</li> <li>• <b>An:</b> ErrorStatus enumeration value:           <ul style="list-style-type: none"> <li>– SUCCESS: Max frequency configuration done</li> <li>– ERROR: Max frequency configuration not done</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                 |
| <p>Notes</p>         | <ul style="list-style-type: none"> <li>• The application need to ensure that PLL is disabled.</li> <li>• Function is based on the following formula: PLL output frequency = ((HSI frequency * PLLMul) / PLLDiv)PLLMul: The application software must set correctly the PLL multiplication factor to avoid exceeding 96 MHz as PLLVCO when the product is in range 1,48 MHz as PLLVCO when the product is in range 2,24 MHz when the product is in range 3</li> <li>• FLASH latency can be modified through this function.</li> <li>• If this latency increases to 1WS, FLASH 64-bit access will be automatically enabled. A decrease of FLASH latency to 0WS will not disable 64-bit access. If needed, user should call the following function LL_FLASH_Disable64bitAccess.</li> </ul> |

### **LL\_PLL\_ConfigSystemClock\_HSE**

- |                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Function name</p>        | <b>ErrorStatus LL_PLL_ConfigSystemClock_HSE (uint32_t HSEFrequency, uint32_t HSEBypass, LL_UTILS_PLLInitTypeDef * UTILS_PLLInitStruct, LL_UTILS_ClkInitTypeDef * UTILS_ClkInitStruct)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <p>Function description</p> | This function configures system clock with HSE as clock source of the PLL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <p>Parameters</p>           | <ul style="list-style-type: none"> <li>• <b>HSEFrequency:</b> Value between Min_Data = 1000000 and Max_Data = 24000000</li> <li>• <b>HSEBypass:</b> This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– LL_UTILS_HSEBYPASS_ON</li> <li>– LL_UTILS_HSEBYPASS_OFF</li> </ul> </li> <li>• <b>UTILS_PLLInitStruct:</b> pointer to a LL_UTILS_PLLInitTypeDef structure that contains the configuration information for the PLL.</li> <li>• <b>UTILS_ClkInitStruct:</b> pointer to a LL_UTILS_ClkInitTypeDef structure that contains the configuration information for the BUS prescalers.</li> </ul> |
| <p>Return values</p>        | <ul style="list-style-type: none"> <li>• <b>An:</b> ErrorStatus enumeration value:           <ul style="list-style-type: none"> <li>– SUCCESS: Max frequency configuration done</li> <li>– ERROR: Max frequency configuration not done</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                     |
| <p>Notes</p>                | <ul style="list-style-type: none"> <li>• The application need to ensure that PLL is disabled.</li> <li>• Function is based on the following formula: PLL output frequency = ((HSE frequency * PLLMul) / PLLDiv)PLLMul: The application software must set correctly the PLL multiplication factor to avoid exceeding 96 MHz as PLLVCO when the product is in range 1,48 MHz as PLLVCO when the product is in range 2,24 MHz when the product is in range 3</li> <li>• FLASH latency can be modified through this function.</li> <li>• If this latency increases to 1WS, FLASH 64-bit access will be</li> </ul>                                     |

automatically enabled. A decrease of FLASH latency to 0WS will not disable 64-bit access. If needed, user should call the following function LL\_FLASH\_Disable64bitAccess.

## 68.3 UTILS Firmware driver defines

### 68.3.1 UTILS

#### *HSE Bypass activation*

`LL_UTILS_HSEBYPASS_OFF` HSE Bypass is not enabled

`LL_UTILS_HSEBYPASS_ON` HSE Bypass is enabled

## 69 LL WWDG Generic Driver

### 69.1 WWDG Firmware driver API description

#### 69.1.1 Detailed description of functions

##### **LL\_WWDG\_Enable**

|                                                   |                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_WWDG_Enable (WWDG_TypeDef * WWDGx)</code>                                                                                                                                                                                                                          |
| Function description                              | Enable Window Watchdog.                                                                                                                                                                                                                                                                          |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> </ul>                                                                                                                                                                                                                  |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                 |
| Notes                                             | <ul style="list-style-type: none"> <li>• It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset. This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR WDGA LL_WWDG_Enable</li> </ul>                                                                                                                                                                                                                       |

##### **LL\_WWDG\_IsEnabled**

|                                                   |                                                                                    |
|---------------------------------------------------|------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_WWDG_IsEnabled (WWDG_TypeDef * WWDGx)</code>     |
| Function description                              | Checks if Window Watchdog is enabled.                                              |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> </ul>    |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR WDGA LL_WWDG_IsEnabled</li> </ul>      |

##### **LL\_WWDG\_SetCounter**

|                      |                                                                                                                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE void LL_WWDG_SetCounter (WWDG_TypeDef * WWDGx, uint32_t Counter)</code>                                                                                                                                                                                                                   |
| Function description | Set the Watchdog counter value to provided value (7-bits T[6:0])                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> <li>• <b>Counter:</b> 0..0x7F (7 bit counter value)</li> </ul>                                                                                                                                                                        |
| Return values        | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• When writing to the WWDG_CR register, always write 1 in the MSB b6 to avoid generating an immediate reset. This counter is decremented every (4096 x 2<sup>exp(WDGTB)</sup>) PCLK cycles. A reset is produced when it rolls over from 0x40 to 0x3F (bit T6)</li> </ul> |

|                                                   |                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------|
|                                                   | becomes cleared) Setting the counter lower then 0x40 causes an immediate reset (if WWDG enabled) |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR T LL_WWDG_SetCounter</li> </ul>                      |

### LL\_WWDG\_GetCounter

|                                                   |                                                                                            |
|---------------------------------------------------|--------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE uint32_t LL_WWDG_GetCounter(<br/>(WWDG_TypeDef * WWDGx)</code></b> |
| Function description                              | Return current Watchdog Counter Value (7 bits counter value)                               |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> </ul>            |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>7:</b> bit Watchdog Counter value</li> </ul>   |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CR T LL_WWDG_GetCounter</li> </ul>                |

### LL\_WWDG\_SetPrescaler

|                                                   |                                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <b><code>_STATIC_INLINE void LL_WWDG_SetPrescaler(<br/>(WWDG_TypeDef * WWDGx, uint32_t Prescaler)</code></b>                                                                                                                                                                                                                          |
| Function description                              | Set the time base of the prescaler (WDGTB).                                                                                                                                                                                                                                                                                           |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> <li>• <b>Prescaler:</b> This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– LL_WWDG_PRESCALER_1</li> <li>– LL_WWDG_PRESCALER_2</li> <li>– LL_WWDG_PRESCALER_4</li> <li>– LL_WWDG_PRESCALER_8</li> </ul> </li> </ul> |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                      |
| Notes                                             | <ul style="list-style-type: none"> <li>• Prescaler is used to apply ratio on PCLK clock, so that Watchdog counter is decremented every (4096 x 2<sup>expWDGTB</sup>) PCLK cycles</li> </ul>                                                                                                                                           |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CFR WDGTB LL_WWDG_SetPrescaler</li> </ul>                                                                                                                                                                                                                                                    |

### LL\_WWDG\_GetPrescaler

|                      |                                                                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name        | <b><code>_STATIC_INLINE uint32_t LL_WWDG_GetPrescaler(<br/>(WWDG_TypeDef * WWDGx)</code></b>                                                                                                                                                          |
| Function description | Return current Watchdog Prescaler Value.                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> </ul>                                                                                                                                                                       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Returned:</b> value can be one of the following values: <ul style="list-style-type: none"> <li>– LL_WWDG_PRESCALER_1</li> <li>– LL_WWDG_PRESCALER_2</li> <li>– LL_WWDG_PRESCALER_4</li> </ul> </li> </ul> |

- LL\_WWDG\_PRESCALER\_8
  - CFR WDGTB LL\_WWDG\_GetPrescaler
- Reference Manual to  
LL API cross  
reference:

### **LL\_WWDG\_SetWindow**

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE void LL_WWDG_SetWindow(<br/>    WWDG_TypeDef * WWDGx, uint32_t Window)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Function description                              | Set the Watchdog Window value to be compared to the downcounter (7-bits W[6:0]).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> <li>• <b>Window:</b> 0x00..0x7F (7 bit Window value)</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>None:</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Notes                                             | <ul style="list-style-type: none"> <li>• This window value defines when write in the WWDG_CR register to program Watchdog counter is allowed. Watchdog counter value update must occur only when the counter value is lower than the Watchdog window register value. Otherwise, a MCU reset is generated if the 7-bit Watchdog counter value (in the control register) is refreshed before the downcounter has reached the watchdog window register value. Physically is possible to set the Window lower than 0x40 but it is not recommended. To generate an immediate reset, it is possible to set the Counter lower than 0x40.</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CFR W LL_WWDG_SetWindow</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

### **LL\_WWDG\_GetWindow**

|                                                   |                                                                                         |
|---------------------------------------------------|-----------------------------------------------------------------------------------------|
| Function name                                     | <code>__STATIC_INLINE uint32_t LL_WWDG_GetWindow(<br/>    WWDG_TypeDef * WWDGx)</code>  |
| Function description                              | Return current Watchdog Window Value (7 bits value)                                     |
| Parameters                                        | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> </ul>         |
| Return values                                     | <ul style="list-style-type: none"> <li>• <b>7:</b> bit Watchdog Window value</li> </ul> |
| Reference Manual to<br>LL API cross<br>reference: | <ul style="list-style-type: none"> <li>• CFR W LL_WWDG_SetWindow</li> </ul>             |

### **LL\_WWDG\_IsActiveFlag\_EWKUP**

|                      |                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------|
| Function name        | <code>__STATIC_INLINE uint32_t LL_WWDG_IsActiveFlag_EWKUP(<br/>    WWDG_TypeDef * WWDGx)</code>                  |
| Function description | Indicates if the WWDG Early Wakeup Interrupt Flag is set or not.                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>WWDGx:</b> WWDG Instance</li> </ul>                                  |
| Return values        | <ul style="list-style-type: none"> <li>• <b>State:</b> of bit (1 or 0).</li> </ul>                               |
| Notes                | <ul style="list-style-type: none"> <li>• This bit is set by hardware when the counter has reached the</li> </ul> |

value 0x40. It must be cleared by software by writing 0. A write of 1 has no effect. This bit is also set if the interrupt is not enabled.

Reference Manual to  
LL API cross  
reference:

- SR EWIF LL\_WWDG\_IsActiveFlag\_EWKUP

### LL\_WWDG\_ClearFlag\_EWKUP

Function name `__STATIC_INLINE void LL_WWDG_ClearFlag_EWKUP (WWDG_TypeDef * WWDGx)`

Function description Clear WWDG Early Wakeup Interrupt Flag (EWIF)

Parameters • **WWDGx:** WWDG Instance

Return values • **None:**

Reference Manual to  
LL API cross  
reference:

- SR EWIF LL\_WWDG\_ClearFlag\_EWKUP

### LL\_WWDG\_EnableIT\_EWKUP

Function name `__STATIC_INLINE void LL_WWDG_EnableIT_EWKUP (WWDG_TypeDef * WWDGx)`

Function description Enable the Early Wakeup Interrupt.

Parameters • **WWDGx:** WWDG Instance

Return values • **None:**

Notes • When set, an interrupt occurs whenever the counter reaches value 0x40. This interrupt is only cleared by hardware after a reset

Reference Manual to  
LL API cross  
reference:

- CFR EWI LL\_WWDG\_EnableIT\_EWKUP

### LL\_WWDG\_IsEnabledIT\_EWKUP

Function name `__STATIC_INLINE uint32_t LL_WWDG_IsEnabledIT_EWKUP (WWDG_TypeDef * WWDGx)`

Function description Check if Early Wakeup Interrupt is enabled.

Parameters • **WWDGx:** WWDG Instance

Return values • **State:** of bit (1 or 0).

Reference Manual to  
LL API cross  
reference:

- CFR EWI LL\_WWDG\_IsEnabledIT\_EWKUP

## 69.2 WWDG Firmware driver defines

### 69.2.1 WWDG

#### *IT Defines*

`LL_WWDG_CFR_EWI`

#### **PRESCALER**

`LL_WWDG_PRESCALER_1` WWDG counter clock = (PCLK1/4096)/1

`LL_WWDG_PRESCALER_2` WWDG counter clock = (PCLK1/4096)/2

`LL_WWDG_PRESCALER_4` WWDG counter clock = (PCLK1/4096)/4

`LL_WWDG_PRESCALER_8` WWDG counter clock = (PCLK1/4096)/8

#### **Common Write and read registers macros**

`LL_WWDG_WriteReg` **Description:**

- Write a value in WWDG register.

#### **Parameters:**

- `_INSTANCE_`: WWDG Instance
- `_REG_`: Register to be written
- `_VALUE_`: Value to be written in the register

#### **Return value:**

- None

`LL_WWDG_ReadReg` **Description:**

- Read a value in WWDG register.

#### **Parameters:**

- `_INSTANCE_`: WWDG Instance
- `_REG_`: Register to be read

#### **Return value:**

- Register: value

## 70 Correspondence between API registers and API low-layer driver functions

### 70.1 ADC

Table 25: Correspondence between ADC registers and ADC low-layer driver functions

| Register | Field   | Function                                  |
|----------|---------|-------------------------------------------|
| CCR      | ADCPRE  | <i>LL_ADC_GetCommonClock</i>              |
|          |         | <i>LL_ADC_SetCommonClock</i>              |
|          | TSVREFE | <i>LL_ADC_GetCommonPathInternalCh</i>     |
|          |         | <i>LL_ADC_SetCommonPathInternalCh</i>     |
|          | AWD1CH  | <i>LL_ADC_GetAnalogWDMonitChannels</i>    |
|          |         | <i>LL_ADC_SetAnalogWDMonitChannels</i>    |
|          | AWD1EN  | <i>LL_ADC_GetAnalogWDMonitChannels</i>    |
|          |         | <i>LL_ADC_SetAnalogWDMonitChannels</i>    |
| CR1      | AWD1SGL | <i>LL_ADC_GetAnalogWDMonitChannels</i>    |
|          |         | <i>LL_ADC_SetAnalogWDMonitChannels</i>    |
|          | AWDIE   | <i>LL_ADC_EnableIT_AWD1</i>               |
|          | DISCEN  | <i>LL_ADC_INJ_SetSequencerDiscont</i>     |
|          |         | <i>LL_ADC_REG_GetSequencerDiscont</i>     |
|          |         | <i>LL_ADC_REG_SetSequencerDiscont</i>     |
|          | DISCNUM | <i>LL_ADC_REG_GetSequencerDiscont</i>     |
|          |         | <i>LL_ADC_REG_SetSequencerDiscont</i>     |
|          | EOCIE   | <i>LL_ADC_DisableIT_EOCS</i>              |
|          |         | <i>LL_ADC_EnableIT_EOCS</i>               |
|          |         | <i>LL_ADC_IsEnabledIT_EOCS</i>            |
|          | JAUTO   | <i>LL_ADC_INJ_GetTrigAuto</i>             |
|          |         | <i>LL_ADC_INJ_SetTrigAuto</i>             |
|          | JEOCIE  | <i>LL_ADC_EnableIT_JEOS</i>               |
|          | OVRIE   | <i>LL_ADC_DisableIT_OVR</i>               |
|          |         | <i>LL_ADC_EnableIT_OVR</i>                |
|          |         | <i>LL_ADC_IsEnabledIT_OVR</i>             |
|          | PDD     | <i>LL_ADC_GetLowPowerModeAutoPowerOff</i> |
|          | PDI     | <i>LL_ADC_SetLowPowerModeAutoPowerOff</i> |
|          | RES     | <i>LL_ADC_GetResolution</i>               |
|          |         | <i>LL_ADC_SetResolution</i>               |
|          | SCAN    | <i>LL_ADC_GetSequencersScanMode</i>       |

| Register | Field    | Function                                 |
|----------|----------|------------------------------------------|
| CR2      | ADC_CFG  | <i>LL_ADC_SetSequencersScanMode</i>      |
|          |          | <i>LL_ADC_GetChannelsBank</i>            |
|          | ADON     | <i>LL_ADC_SetChannelsBank</i>            |
|          |          | <i>LL_ADC_Disable</i>                    |
|          |          | <i>LL_ADC_Enable</i>                     |
|          | ALIGN    | <i>LL_ADC_IsEnabled</i>                  |
|          |          | <i>LL_ADC_SetDataAlignment</i>           |
|          | CONT     | <i>LL_ADC_REG_GetContinuousMode</i>      |
|          |          | <i>LL_ADC_REG_SetContinuousMode</i>      |
|          | DDS      | <i>LL_ADC_REG_GetDMATransfer</i>         |
|          |          | <i>LL_ADC_REG_SetDMATransfer</i>         |
|          | DELS     | <i>LL_ADC_GetLowPowerModeAutoWait</i>    |
|          |          | <i>LL_ADC_SetLowPowerModeAutoWait</i>    |
|          | DMA      | <i>LL_ADC_REG_GetDMATransfer</i>         |
|          |          | <i>LL_ADC_REG_SetDMATransfer</i>         |
|          | EOCS     | <i>LL_ADC_REG_GetFlagEndOfConversion</i> |
|          |          | <i>LL_ADC_REG_SetFlagEndOfConversion</i> |
|          | EXTEN    | <i>LL_ADC_REG_GetTriggerEdge</i>         |
|          |          | <i>LL_ADC_REG_GetTriggerSource</i>       |
|          |          | <i>LL_ADC_REG_IsTriggerSourceSWStart</i> |
|          |          | <i>LL_ADC_REG_SetTriggerSource</i>       |
|          |          | <i>LL_ADC_REG_StartConversionExtTrig</i> |
|          |          | <i>LL_ADC_REG_StopConversionExtTrig</i>  |
|          | EXTSEL   | <i>LL_ADC_REG_GetTriggerSource</i>       |
|          |          | <i>LL_ADC_REG_SetTriggerSource</i>       |
|          | JEXTEN   | <i>LL_ADC_INJ_GetTriggerEdge</i>         |
|          |          | <i>LL_ADC_INJ_GetTriggerSource</i>       |
|          |          | <i>LL_ADC_INJ_IsTriggerSourceSWStart</i> |
|          |          | <i>LL_ADC_INJ_SetTriggerSource</i>       |
|          |          | <i>LL_ADC_INJ_StartConversionExtTrig</i> |
|          |          | <i>LL_ADC_INJ_StopConversionExtTrig</i>  |
|          | JEXTSEL  | <i>LL_ADC_INJ_GetTriggerSource</i>       |
|          |          | <i>LL_ADC_INJ_SetTriggerSource</i>       |
|          | JSWSTART | <i>LL_ADC_INJ_StartConversionSWStart</i> |
|          | SWSTART  | <i>LL_ADC_REG_StartConversionSWStart</i> |
| CSR      | FCH3     | <i>LL_ADC_GetChannelRouting</i>          |

| Register | Field    | Function                               |
|----------|----------|----------------------------------------|
| DR       | FCH8     | <i>LL_ADC_SetChannelRouting</i>        |
|          |          | <i>LL_ADC_GetChannelRouting</i>        |
|          |          | <i>LL_ADC_SetChannelRouting</i>        |
|          | RCH13    | <i>LL_ADC_GetChannelRouting</i>        |
|          |          | <i>LL_ADC_SetChannelRouting</i>        |
|          |          |                                        |
| HTR      | DATA     | <i>LL_ADC_DMA_GetRegAddr</i>           |
|          |          | <i>LL_ADC_REG_ReadConversionData10</i> |
|          |          | <i>LL_ADC_REG_ReadConversionData12</i> |
|          |          | <i>LL_ADC_REG_ReadConversionData32</i> |
|          |          | <i>LL_ADC_REG_ReadConversionData6</i>  |
|          |          | <i>LL_ADC_REG_ReadConversionData8</i>  |
| JDR1     | HT       | <i>LL_ADC_GetAnalogWDThresholds</i>    |
|          |          | <i>LL_ADC_SetAnalogWDThresholds</i>    |
| JDR2     | JDATA    | <i>LL_ADC_INJ_ReadConversionData10</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData12</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData32</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData6</i>  |
|          |          | <i>LL_ADC_INJ_ReadConversionData8</i>  |
| JDR3     | JDATA    | <i>LL_ADC_INJ_ReadConversionData10</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData12</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData32</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData6</i>  |
|          |          | <i>LL_ADC_INJ_ReadConversionData8</i>  |
| JDR4     | JDATA    | <i>LL_ADC_INJ_ReadConversionData10</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData12</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData32</i> |
|          |          | <i>LL_ADC_INJ_ReadConversionData6</i>  |
|          |          | <i>LL_ADC_INJ_ReadConversionData8</i>  |
| JOFR1    | JOFFSET1 | <i>LL_ADC_INJ_GetOffset</i>            |
|          |          | <i>LL_ADC_INJ_SetOffset</i>            |
| JOFR2    | JOFFSET2 | <i>LL_ADC_INJ_GetOffset</i>            |

| Register | Field    | Function                             |
|----------|----------|--------------------------------------|
| JOFR3    | JOFFSET3 | <i>LL_ADC_INJ_SetOffset</i>          |
|          |          | <i>LL_ADC_INJ_GetOffset</i>          |
|          |          | <i>LL_ADC_INJ_SetOffset</i>          |
| JOFR4    | JOFFSET4 | <i>LL_ADC_INJ_GetOffset</i>          |
|          |          | <i>LL_ADC_INJ_SetOffset</i>          |
|          |          | <i>LL_ADC_INJ_SetOffset</i>          |
| JSQR     | JL       | <i>LL_ADC_INJ_GetSequencerLength</i> |
|          |          | <i>LL_ADC_INJ_SetSequencerLength</i> |
|          | JSQ1     | <i>LL_ADC_INJ_SetSequencerRanks</i>  |
|          | JSQ2     | <i>LL_ADC_INJ_SetSequencerRanks</i>  |
|          | JSQ3     | <i>LL_ADC_INJ_SetSequencerRanks</i>  |
|          | JSQ4     | <i>LL_ADC_INJ_SetSequencerRanks</i>  |
| LTR      | LT       | <i>LL_ADC_GetAnalogWDThresholds</i>  |
|          |          | <i>LL_ADC_SetAnalogWDThresholds</i>  |
| SMPR0    | SMP30    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP31    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
| SMPR1    | SMP20    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP21    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP22    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP23    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP24    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP25    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP26    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP27    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP28    | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |          | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP29    | <i>LL_ADC_GetChannelSamplingTime</i> |

| Register | Field | Function                             |
|----------|-------|--------------------------------------|
| SMPR2    |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP10 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP11 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP12 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP13 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP14 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP15 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP16 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP17 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP18 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP19 | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
| SMPR3    | SMP0  | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP1  | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP2  | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP3  | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
| SMPR4    | SMP4  | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP5  | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
| SMPR5    | SMP6  | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP7  | <i>LL_ADC_GetChannelSamplingTime</i> |

| Register | Field | Function                             |
|----------|-------|--------------------------------------|
| SMP8     |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_GetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
|          | SMP9  | <i>LL_ADC_SetChannelSamplingTime</i> |
|          |       | <i>LL_ADC_SetChannelSamplingTime</i> |
| SQR1     | L     | <i>LL_ADC_REG_SetSequencerLength</i> |
|          | SQ25  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ26  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ27  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ28  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
| SQR2     | SQ19  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ20  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ21  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ22  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ23  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
| SQR3     | SQ24  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ13  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ14  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ15  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
| SQ16     | SQ16  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |
|          | SQ17  | <i>LL_ADC_REG_GetSequencerRanks</i>  |
|          |       | <i>LL_ADC_REG_SetSequencerRanks</i>  |

| Register | Field | Function                                            |
|----------|-------|-----------------------------------------------------|
| SQR4     | SQ18  | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ10  | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ11  | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ12  | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ7   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
| SQR5     | SQ8   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ9   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ1   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ2   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ3   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ4   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ5   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
|          | SQ6   | <a href="#"><i>LL_ADC_REG_GetSequencerRanks</i></a> |
|          |       | <a href="#"><i>LL_ADC_REG_SetSequencerRanks</i></a> |
| SR       | ADONS | <a href="#"><i>LL_ADC_IsActiveFlag_ADRDY</i></a>    |
|          | AWD   | <a href="#"><i>LL_ADC_ClearFlag_AWD1</i></a>        |
|          |       | <a href="#"><i>LL_ADC_IsActiveFlag_AWD1</i></a>     |
|          | EOC   | <a href="#"><i>LL_ADC_ClearFlag_EOCS</i></a>        |
|          |       | <a href="#"><i>LL_ADC_IsActiveFlag_EOCS</i></a>     |
|          | JEOC  | <a href="#"><i>LL_ADC_ClearFlag_JEOS</i></a>        |
|          |       | <a href="#"><i>LL_ADC_IsActiveFlag_JEOS</i></a>     |
|          | OVR   | <a href="#"><i>LL_ADC_ClearFlag_OVR</i></a>         |
|          |       | <a href="#"><i>LL_ADC_IsActiveFlag_OVR</i></a>      |

## 70.2 BUS

**Table 26: Correspondence between BUS registers and BUS low-layer driver functions**

| Register | Field   | Function                           |
|----------|---------|------------------------------------|
| AHBENR   | AESEN   | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | CRCEN   | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | DMA1EN  | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | DMA2EN  | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | FLITFEN | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | FSMCEN  | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | GPIOAEN | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | GPIOBEN | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | GPIOCEN | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | GPIODEN | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | GPIOEEN | <i>LL_AHB1_GRP1_DisableClock</i>   |
|          |         | <i>LL_AHB1_GRP1_EnableClock</i>    |
|          |         | <i>LL_AHB1_GRP1_IsEnabledClock</i> |
|          | GPIOFEN | <i>LL_AHB1_GRP1_DisableClock</i>   |

| Register | Field     | Function                              |
|----------|-----------|---------------------------------------|
| AHBLPENR | GPIOGEN   | <i>LL_AHB1_GRP1_EnableClock</i>       |
|          |           | <i>LL_AHB1_GRP1_IsEnabledClock</i>    |
|          |           | <i>LL_AHB1_GRP1_DisableClock</i>      |
|          |           | <i>LL_AHB1_GRP1_EnableClock</i>       |
|          | GPIOHEN   | <i>LL_AHB1_GRP1_IsEnabledClock</i>    |
|          |           | <i>LL_AHB1_GRP1_DisableClock</i>      |
|          |           | <i>LL_AHB1_GRP1_EnableClock</i>       |
|          |           | <i>LL_AHB1_GRP1_IsEnabledClock</i>    |
|          | AESLPEN   | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | CRCLPEN   | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | DMA1LPEN  | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | DMA2LPEN  | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | FLITFLPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | FSMCLPEN  | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | GPIOALPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | GPIOBLPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | GPIOCLPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | GPIODLPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | GPIOELPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | GPIOFLPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | GPIOGLPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |
|          | GPIOHLPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i> |
|          |           | <i>LL_AHB1_GRP1_EnableClockSleep</i>  |

| Register | Field    | Function                                                                                                  |
|----------|----------|-----------------------------------------------------------------------------------------------------------|
|          | SRAMLPEN | <i>LL_AHB1_GRP1_DisableClockSleep</i><br><i>LL_AHB1_GRP1_EnableClockSleep</i>                             |
|          | AESRST   | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | CRCRST   | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | DMA1RST  | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | DMA2RST  | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | FLITFRST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | FSMCRST  | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
| AHBRSTR  | GPIOARST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | GPIOBRST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | GPIOCRST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | GPIODRST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | GPIOERST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | GPIOFRST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | GPIOGRST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
|          | GPIOHRST | <i>LL_AHB1_GRP1_ForceReset</i><br><i>LL_AHB1_GRP1_ReleaseReset</i>                                        |
| APB1ENR  | COMPEN   | <i>LL_APB1_GRP1_DisableClock</i><br><i>LL_APB1_GRP1_EnableClock</i><br><i>LL_APB1_GRP1_IsEnabledClock</i> |
|          |          | <i>LL_APB1_GRP1_DisableClock</i><br><i>LL_APB1_GRP1_EnableClock</i><br><i>LL_APB1_GRP1_IsEnabledClock</i> |
|          |          | <i>LL_APB1_GRP1_DisableClock</i><br><i>LL_APB1_GRP1_EnableClock</i><br><i>LL_APB1_GRP1_IsEnabledClock</i> |

| Register | Field  | Function                           |
|----------|--------|------------------------------------|
|          | I2C1EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | I2C2EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | LCDEN  | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | PWREN  | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | SPI2EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | SPI3EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | TIM2EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | TIM3EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | TIM4EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | TIM5EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | TIM6EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |
|          | TIM7EN | <i>LL_APB1_GRP1_DisableClock</i>   |
|          |        | <i>LL_APB1_GRP1_EnableClock</i>    |
|          |        | <i>LL_APB1_GRP1_IsEnabledClock</i> |

| Register | Field    | Function                              |
|----------|----------|---------------------------------------|
|          | UART4EN  | <i>LL_APB1_GRP1_DisableClock</i>      |
|          |          | <i>LL_APB1_GRP1_EnableClock</i>       |
|          |          | <i>LL_APB1_GRP1_IsEnabledClock</i>    |
|          | UART5EN  | <i>LL_APB1_GRP1_DisableClock</i>      |
|          |          | <i>LL_APB1_GRP1_EnableClock</i>       |
|          |          | <i>LL_APB1_GRP1_IsEnabledClock</i>    |
|          | USART2EN | <i>LL_APB1_GRP1_DisableClock</i>      |
|          |          | <i>LL_APB1_GRP1_EnableClock</i>       |
|          |          | <i>LL_APB1_GRP1_IsEnabledClock</i>    |
|          | USART3EN | <i>LL_APB1_GRP1_DisableClock</i>      |
|          |          | <i>LL_APB1_GRP1_EnableClock</i>       |
|          |          | <i>LL_APB1_GRP1_IsEnabledClock</i>    |
|          | USBEN    | <i>LL_APB1_GRP1_DisableClock</i>      |
|          |          | <i>LL_APB1_GRP1_EnableClock</i>       |
|          |          | <i>LL_APB1_GRP1_IsEnabledClock</i>    |
|          | WWDGEN   | <i>LL_APB1_GRP1_DisableClock</i>      |
|          |          | <i>LL_APB1_GRP1_EnableClock</i>       |
|          |          | <i>LL_APB1_GRP1_IsEnabledClock</i>    |
|          | COMPLPEN | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | DACLPEN  | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | I2C1LPEN | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | I2C2LPEN | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | LCDLPEN  | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | PWRLPEN  | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | SPI2LPEN | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | SPI3LPEN | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | TIM2LPEN | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |          | <i>LL_APB1_GRP1_EnableClockSleep</i>  |

| Register | Field      | Function                              |
|----------|------------|---------------------------------------|
|          | TIM3LPEN   | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | TIM4LPEN   | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | TIM5LPEN   | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | TIM6LPEN   | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | TIM7LPEN   | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | UART4LPEN  | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | UART5LPEN  | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | USART2LPEN | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | USART3LPEN | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | USBLPEN    | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
|          | WWDGLPEN   | <i>LL_APB1_GRP1_DisableClockSleep</i> |
|          |            | <i>LL_APB1_GRP1_EnableClockSleep</i>  |
| APB1RSTR | COMPRST    | <i>LL_APB1_GRP1_ForceReset</i>        |
|          |            | <i>LL_APB1_GRP1_ReleaseReset</i>      |
|          | DACRST     | <i>LL_APB1_GRP1_ForceReset</i>        |
|          |            | <i>LL_APB1_GRP1_ReleaseReset</i>      |
|          | I2C1RST    | <i>LL_APB1_GRP1_ForceReset</i>        |
|          |            | <i>LL_APB1_GRP1_ReleaseReset</i>      |
|          | I2C2RST    | <i>LL_APB1_GRP1_ForceReset</i>        |
|          |            | <i>LL_APB1_GRP1_ReleaseReset</i>      |
|          | LCDRST     | <i>LL_APB1_GRP1_ForceReset</i>        |
|          |            | <i>LL_APB1_GRP1_ReleaseReset</i>      |
|          | PWRRST     | <i>LL_APB1_GRP1_ForceReset</i>        |
|          |            | <i>LL_APB1_GRP1_ReleaseReset</i>      |
|          | SPI2RST    | <i>LL_APB1_GRP1_ForceReset</i>        |
|          |            | <i>LL_APB1_GRP1_ReleaseReset</i>      |

| Register | Field     | Function                           |
|----------|-----------|------------------------------------|
|          | SPI3RST   | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | TIM2RST   | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | TIM3RST   | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | TIM4RST   | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | TIM5RST   | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | TIM6RST   | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | TIM7RST   | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | UART4RST  | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | UART5RST  | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | USART2RST | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | USART3RST | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | USBRST    | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
|          | WWDGRST   | <i>LL_APB1_GRP1_ForceReset</i>     |
|          |           | <i>LL_APB1_GRP1_ReleaseReset</i>   |
| APB2ENR  | ADC1EN    | <i>LL_APB2_GRP1_DisableClock</i>   |
|          |           | <i>LL_APB2_GRP1_EnableClock</i>    |
|          |           | <i>LL_APB2_GRP1_IsEnabledClock</i> |
|          | SDIOEN    | <i>LL_APB2_GRP1_DisableClock</i>   |
|          |           | <i>LL_APB2_GRP1_EnableClock</i>    |
|          |           | <i>LL_APB2_GRP1_IsEnabledClock</i> |
|          | SPI1EN    | <i>LL_APB2_GRP1_DisableClock</i>   |
|          |           | <i>LL_APB2_GRP1_EnableClock</i>    |
|          |           | <i>LL_APB2_GRP1_IsEnabledClock</i> |
|          | SYSCFGEN  | <i>LL_APB2_GRP1_DisableClock</i>   |

| Register  | Field      | Function                              |
|-----------|------------|---------------------------------------|
| APB2LPENR | TIM10EN    | <i>LL_APB2_GRP1_DisableClock</i>      |
|           |            | <i>LL_APB2_GRP1_IsEnabledClock</i>    |
|           | TIM11EN    | <i>LL_APB2_GRP1_DisableClock</i>      |
|           |            | <i>LL_APB2_GRP1_EnableClock</i>       |
|           |            | <i>LL_APB2_GRP1_IsEnabledClock</i>    |
|           | TIM9EN     | <i>LL_APB2_GRP1_DisableClock</i>      |
|           |            | <i>LL_APB2_GRP1_EnableClock</i>       |
|           |            | <i>LL_APB2_GRP1_IsEnabledClock</i>    |
|           | USART1EN   | <i>LL_APB2_GRP1_DisableClock</i>      |
|           |            | <i>LL_APB2_GRP1_EnableClock</i>       |
|           |            | <i>LL_APB2_GRP1_IsEnabledClock</i>    |
| APB2RSTR  | ADC1LPEN   | <i>LL_APB2_GRP1_DisableClockSleep</i> |
|           |            | <i>LL_APB2_GRP1_EnableClockSleep</i>  |
|           | SDIOLPEN   | <i>LL_APB2_GRP1_DisableClockSleep</i> |
|           |            | <i>LL_APB2_GRP1_EnableClockSleep</i>  |
|           | SPI1LPEN   | <i>LL_APB2_GRP1_DisableClockSleep</i> |
|           |            | <i>LL_APB2_GRP1_EnableClockSleep</i>  |
|           | SYSCFGLPEN | <i>LL_APB2_GRP1_DisableClockSleep</i> |
|           |            | <i>LL_APB2_GRP1_EnableClockSleep</i>  |
|           | TIM10LPEN  | <i>LL_APB2_GRP1_DisableClockSleep</i> |
|           |            | <i>LL_APB2_GRP1_EnableClockSleep</i>  |
|           | TIM11LPEN  | <i>LL_APB2_GRP1_DisableClockSleep</i> |
|           |            | <i>LL_APB2_GRP1_EnableClockSleep</i>  |
|           | TIM9LPEN   | <i>LL_APB2_GRP1_DisableClockSleep</i> |
|           |            | <i>LL_APB2_GRP1_EnableClockSleep</i>  |
|           | USART1LPEN | <i>LL_APB2_GRP1_DisableClockSleep</i> |
|           |            | <i>LL_APB2_GRP1_EnableClockSleep</i>  |
| APB2RSTR  | ADC1RST    | <i>LL_APB2_GRP1_ForceReset</i>        |
|           |            | <i>LL_APB2_GRP1_ReleaseReset</i>      |
|           | SDIORST    | <i>LL_APB2_GRP1_ForceReset</i>        |
|           |            | <i>LL_APB2_GRP1_ReleaseReset</i>      |
|           | SPI1RST    | <i>LL_APB2_GRP1_ForceReset</i>        |
|           |            | <i>LL_APB2_GRP1_ReleaseReset</i>      |

| Register | Field     | Function                         |
|----------|-----------|----------------------------------|
|          | SYSCFGRST | <i>LL_APB2_GRP1_ForceReset</i>   |
|          |           | <i>LL_APB2_GRP1_ReleaseReset</i> |
|          | TIM10RST  | <i>LL_APB2_GRP1_ForceReset</i>   |
|          |           | <i>LL_APB2_GRP1_ReleaseReset</i> |
|          | TIM11RST  | <i>LL_APB2_GRP1_ForceReset</i>   |
|          |           | <i>LL_APB2_GRP1_ReleaseReset</i> |
|          | TIM9RST   | <i>LL_APB2_GRP1_ForceReset</i>   |
|          |           | <i>LL_APB2_GRP1_ReleaseReset</i> |
|          | USART1RST | <i>LL_APB2_GRP1_ForceReset</i>   |
|          |           | <i>LL_APB2_GRP1_ReleaseReset</i> |

## 70.3 COMP

Table 27: Correspondence between COMP registers and COMP low-layer driver functions

| Register  | Field          | Function                               |
|-----------|----------------|----------------------------------------|
| COMP2_CSR | SPEED          | <i>LL_COMP_SetPowerMode</i>            |
|           |                | <i>LL_COMP_SetPowerMode</i>            |
|           | 10KPD          | <i>LL_COMP_SetInputPullingResistor</i> |
|           | 10KPU          | <i>LL_COMP_SetInputPullingResistor</i> |
|           | 400KPD         | <i>LL_COMP_SetInputPullingResistor</i> |
|           | 400KPU         | <i>LL_COMP_SetInputPullingResistor</i> |
|           | CMP1OUT        | <i>LL_COMP_ReadOutputLevel</i>         |
|           | CMP2OUT        | <i>LL_COMP_ReadOutputLevel</i>         |
|           | COMP1EN        | <i>LL_COMP_Disable</i>                 |
|           |                | <i>LL_COMP_Enable</i>                  |
|           |                | <i>LL_COMP_IsEnabled</i>               |
| CSR       | COMP_CSR_INSEL | <i>LL_COMP_Disable</i>                 |
|           |                | <i>LL_COMP_Enable</i>                  |
|           |                | <i>LL_COMP_IsEnabled</i>               |
|           |                | <i>LL_COMP_SetInputMinus</i>           |
|           | OUTSEL         | <i>LL_COMP_GetOutputSelection</i>      |
|           |                | <i>LL_COMP_SetOutputSelection</i>      |
|           | WNDWE          | <i>LL_COMP_GetCommonWindowMode</i>     |
|           |                | <i>LL_COMP_SetCommonWindowMode</i>     |
|           | RI_ASCR1_CH    | <i>LL_COMP_SetInputPlus</i>            |
|           |                | <i>LL_COMP_SetInputPlus</i>            |
|           | RI_ASCR2_GR6   | <i>LL_COMP_SetInputPlus</i>            |

| Register | Field | Function                                    |
|----------|-------|---------------------------------------------|
|          |       | <a href="#"><i>LL_COMP_SetInputPlus</i></a> |

## 70.4 CORTEX

Table 28: Correspondence between CORTEX registers and CORTEX low-layer driver functions

| Register  | Field        | Function                                              |
|-----------|--------------|-------------------------------------------------------|
| MPU_CTRL  | ENABLE       | <a href="#"><i>LL_MPUI_Disable</i></a>                |
|           |              | <a href="#"><i>LL_MPUI_Enable</i></a>                 |
|           |              | <a href="#"><i>LL_MPUI_IsEnabled</i></a>              |
| MPU_RASR  | AP           | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
|           | B            | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
|           | C            | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
|           | ENABLE       | <a href="#"><i>LL_MPUI_DisableRegion</i></a>          |
|           |              | <a href="#"><i>LL_MPUI_EnableRegion</i></a>           |
|           | S            | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
|           | SIZE         | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
|           | XN           | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
| MPU_RBAR  | ADDR         | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
|           | REGION       | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
| MPU_RNR   | REGION       | <a href="#"><i>LL_MPUI_ConfigRegion</i></a>           |
|           |              | <a href="#"><i>LL_MPUI_DisableRegion</i></a>          |
| SCB_CPUID | ARCHITECTURE | <a href="#"><i>LL_CPUID_GetConstant</i></a>           |
|           | IMPLEMENTER  | <a href="#"><i>LL_CPUID_GetImplementer</i></a>        |
|           | PARTNO       | <a href="#"><i>LL_CPUID_GetParNo</i></a>              |
|           | REVISION     | <a href="#"><i>LL_CPUID_GetRevision</i></a>           |
|           | VARIANT      | <a href="#"><i>LL_CPUID_GetVariant</i></a>            |
| SCB_SCR   | SEVEONPEND   | <a href="#"><i>LL_LPM_DisableEventOnPend</i></a>      |
|           |              | <a href="#"><i>LL_LPM_EnableEventOnPend</i></a>       |
|           | SLEEPDEEP    | <a href="#"><i>LL_LPM_EnableDeepSleep</i></a>         |
|           |              | <a href="#"><i>LL_LPM_EnableSleep</i></a>             |
| SCB_SHCSR | SLEEPONEXIT  | <a href="#"><i>LL_LPM_DisableSleepOnExit</i></a>      |
|           |              | <a href="#"><i>LL_LPM_EnableSleepOnExit</i></a>       |
|           | MEMFAULTENA  | <a href="#"><i>LL_HANDLER_DisableFault</i></a>        |
| STK_CTRL  | CLKSOURCE    | <a href="#"><i>LL_SYSTICK_GetClkSource</i></a>        |
|           |              | <a href="#"><i>LL_SYSTICK_SetClkSource</i></a>        |
|           | COUNTFLAG    | <a href="#"><i>LL_SYSTICK_IsActiveCounterFlag</i></a> |

| Register | Field   | Function                      |
|----------|---------|-------------------------------|
|          | TICKINT | <i>LL_SYSTICK_DisableIT</i>   |
|          |         | <i>LL_SYSTICK_EnableIT</i>    |
|          |         | <i>LL_SYSTICK_IsEnabledIT</i> |

## 70.5 CRC

Table 29: Correspondence between CRC registers and CRC low-layer driver functions

| Register | Field | Function                              |
|----------|-------|---------------------------------------|
| CR       | RESET | <i>LL_CRC_ResetCRCCalculationUnit</i> |
| DR       | DR    | <i>LL_CRC_FeedData32</i>              |
|          |       | <i>LL_CRC_ReadData32</i>              |
| IDR      | IDR   | <i>LL_CRC_Read_IDR</i>                |
|          |       | <i>LL_CRC_Write_IDR</i>               |

## 70.6 DAC

Table 30: Correspondence between DAC registers and DAC low-layer driver functions

| Register | Field     | Function                          |
|----------|-----------|-----------------------------------|
| CR       | BOFF1     | <i>LL_DAC_GetOutputBuffer</i>     |
|          |           | <i>LL_DAC_SetOutputBuffer</i>     |
|          | BOFF2     | <i>LL_DAC_GetOutputBuffer</i>     |
|          |           | <i>LL_DAC_SetOutputBuffer</i>     |
|          | DMAEN1    | <i>LL_DAC_DisableDMAReq</i>       |
|          |           | <i>LL_DAC_EnableDMAReq</i>        |
|          |           | <i>LL_DAC_IsDMAReqEnabled</i>     |
|          | DMAEN2    | <i>LL_DAC_DisableDMAReq</i>       |
|          |           | <i>LL_DAC_EnableDMAReq</i>        |
|          |           | <i>LL_DAC_IsDMAReqEnabled</i>     |
|          | DMAUDRIE1 | <i>LL_DAC_DisableIT_DMAUDR1</i>   |
|          |           | <i>LL_DAC_EnableIT_DMAUDR1</i>    |
|          |           | <i>LL_DAC_IsEnabledIT_DMAUDR1</i> |
|          | DMAUDRIE2 | <i>LL_DAC_DisableIT_DMAUDR2</i>   |
|          |           | <i>LL_DAC_EnableIT_DMAUDR2</i>    |
|          |           | <i>LL_DAC_IsEnabledIT_DMAUDR2</i> |
|          | EN1       | <i>LL_DAC_Disable</i>             |
|          |           | <i>LL_DAC_Enable</i>              |
|          |           | <i>LL_DAC_IsEnabled</i>           |

| Register | Field    | Function                                    |
|----------|----------|---------------------------------------------|
|          | EN2      | <i>LL_DAC_Disable</i>                       |
|          |          | <i>LL_DAC_Enable</i>                        |
|          |          | <i>LL_DAC_IsEnabled</i>                     |
|          | MAMP1    | <i>LL_DAC_GetWaveNoiseLFSR</i>              |
|          |          | <i>LL_DAC_GetWaveTriangleAmplitude</i>      |
|          |          | <i>LL_DAC_SetWaveNoiseLFSR</i>              |
|          |          | <i>LL_DAC_SetWaveTriangleAmplitude</i>      |
|          | MAMP2    | <i>LL_DAC_GetWaveNoiseLFSR</i>              |
|          |          | <i>LL_DAC_GetWaveTriangleAmplitude</i>      |
|          |          | <i>LL_DAC_SetWaveNoiseLFSR</i>              |
|          |          | <i>LL_DAC_SetWaveTriangleAmplitude</i>      |
|          | TEN1     | <i>LL_DAC_DisableTrigger</i>                |
|          |          | <i>LL_DAC_EnableTrigger</i>                 |
|          |          | <i>LL_DAC_IsTriggerEnabled</i>              |
|          | TEN2     | <i>LL_DAC_DisableTrigger</i>                |
|          |          | <i>LL_DAC_EnableTrigger</i>                 |
|          |          | <i>LL_DAC_IsTriggerEnabled</i>              |
|          | TSEL1    | <i>LL_DAC_GetTriggerSource</i>              |
|          |          | <i>LL_DAC_SetTriggerSource</i>              |
|          | TSEL2    | <i>LL_DAC_GetTriggerSource</i>              |
|          |          | <i>LL_DAC_SetTriggerSource</i>              |
|          | WAVE1    | <i>LL_DAC_GetWaveAutoGeneration</i>         |
|          |          | <i>LL_DAC_SetWaveAutoGeneration</i>         |
|          | WAVE2    | <i>LL_DAC_GetWaveAutoGeneration</i>         |
|          |          | <i>LL_DAC_SetWaveAutoGeneration</i>         |
| DHR12L1  | DACC1DHR | <i>LL_DAC_ConvertData12LeftAligned</i>      |
|          |          | <i>LL_DAC_DMA_GetRegAddr</i>                |
| DHR12L2  | DACC2DHR | <i>LL_DAC_ConvertData12LeftAligned</i>      |
|          |          | <i>LL_DAC_DMA_GetRegAddr</i>                |
| DHR12LD  | DACC1DHR | <i>LL_DAC_ConvertDualData12LeftAligned</i>  |
|          | DACC2DHR | <i>LL_DAC_ConvertDualData12LeftAligned</i>  |
| DHR12R1  | DACC1DHR | <i>LL_DAC_ConvertData12RightAligned</i>     |
|          |          | <i>LL_DAC_DMA_GetRegAddr</i>                |
| DHR12R2  | DACC2DHR | <i>LL_DAC_ConvertData12RightAligned</i>     |
|          |          | <i>LL_DAC_DMA_GetRegAddr</i>                |
| DHR12RD  | DACC1DHR | <i>LL_DAC_ConvertDualData12RightAligned</i> |

| Register | Field    | Function                                    |
|----------|----------|---------------------------------------------|
| DHR8R1   | DACC2DHR | <i>LL_DAC_ConvertDualData12RightAligned</i> |
|          | DACC1DHR | <i>LL_DAC_ConvertData8RightAligned</i>      |
|          |          | <i>LL_DAC_DMA_GetRegAddr</i>                |
| DHR8R2   | DACC2DHR | <i>LL_DAC_ConvertData8RightAligned</i>      |
|          |          | <i>LL_DAC_DMA_GetRegAddr</i>                |
| DHR8RD   | DACC1DHR | <i>LL_DAC_ConvertDualData8RightAligned</i>  |
|          | DACC2DHR | <i>LL_DAC_ConvertDualData8RightAligned</i>  |
| DOR1     | DACC1DOR | <i>LL_DAC_RetrieveOutputData</i>            |
| DOR2     | DACC2DOR | <i>LL_DAC_RetrieveOutputData</i>            |
| SR       | DMAUDR1  | <i>LL_DAC_ClearFlag_DMAUDR1</i>             |
|          |          | <i>LL_DAC_IsActiveFlag_DMAUDR1</i>          |
|          | DMAUDR2  | <i>LL_DAC_ClearFlag_DMAUDR2</i>             |
|          |          | <i>LL_DAC_IsActiveFlag_DMAUDR2</i>          |
| SWTRIGR  | SWTRIG1  | <i>LL_DAC_TrigSWConversion</i>              |
|          | SWTRIG2  | <i>LL_DAC_TrigSWConversion</i>              |

## 70.7 DMA

Table 31: Correspondence between DMA registers and DMA low-layer driver functions

| Register | Field   | Function                               |
|----------|---------|----------------------------------------|
| CCR      | CIRC    | <i>LL_DMA_ConfigTransfer</i>           |
|          |         | <i>LL_DMA_GetMode</i>                  |
|          |         | <i>LL_DMA_SetMode</i>                  |
|          | DIR     | <i>LL_DMA_ConfigTransfer</i>           |
|          |         | <i>LL_DMA_GetDataTransferDirection</i> |
|          |         | <i>LL_DMA_SetDataTransferDirection</i> |
|          | EN      | <i>LL_DMA_DisableChannel</i>           |
|          |         | <i>LL_DMA_EnableChannel</i>            |
|          |         | <i>LL_DMA_IsEnabledChannel</i>         |
|          | HTIE    | <i>LL_DMA_DisableIT_HT</i>             |
|          |         | <i>LL_DMA_EnableIT_HT</i>              |
|          |         | <i>LL_DMA_IsEnabledIT_HT</i>           |
|          | MEM2MEM | <i>LL_DMA_ConfigTransfer</i>           |
|          |         | <i>LL_DMA_GetDataTransferDirection</i> |
|          |         | <i>LL_DMA_SetDataTransferDirection</i> |
|          | MINC    | <i>LL_DMA_ConfigTransfer</i>           |
|          |         | <i>LL_DMA_GetMemoryIncMode</i>         |

| Register | Field | Function                              |
|----------|-------|---------------------------------------|
|          | MSIZE | <i>LL_DMA_SetMemoryIncMode</i>        |
|          |       | <i>LL_DMA_ConfigTransfer</i>          |
|          |       | <i>LL_DMA_GetMemorySize</i>           |
|          |       | <i>LL_DMA_SetMemorySize</i>           |
|          | PINC  | <i>LL_DMA_ConfigTransfer</i>          |
|          |       | <i>LL_DMA_GetPeriphIncMode</i>        |
|          |       | <i>LL_DMA_SetPeriphIncMode</i>        |
|          | PL    | <i>LL_DMA_ConfigTransfer</i>          |
|          |       | <i>LL_DMA_GetChannelPriorityLevel</i> |
|          |       | <i>LL_DMA_SetChannelPriorityLevel</i> |
|          | PSIZE | <i>LL_DMA_ConfigTransfer</i>          |
|          |       | <i>LL_DMA_GetPeriphSize</i>           |
|          |       | <i>LL_DMA_SetPeriphSize</i>           |
|          | TCIE  | <i>LL_DMA_DisableIT_TC</i>            |
|          |       | <i>LL_DMA_EnableIT_TC</i>             |
|          |       | <i>LL_DMA_IsEnabledIT_TC</i>          |
|          | TEIE  | <i>LL_DMA_DisableIT_TE</i>            |
|          |       | <i>LL_DMA_EnableIT_TE</i>             |
|          |       | <i>LL_DMA_IsEnabledIT_TE</i>          |
| CMAR     | MA    | <i>LL_DMA_ConfigAddresses</i>         |
|          |       | <i>LL_DMA_GetM2MDstAddress</i>        |
|          |       | <i>LL_DMA_GetMemoryAddress</i>        |
|          |       | <i>LL_DMA_SetM2MDstAddress</i>        |
|          |       | <i>LL_DMA_SetMemoryAddress</i>        |
| CNDTR    | NDT   | <i>LL_DMA_GetDataLength</i>           |
|          |       | <i>LL_DMA_SetDataLength</i>           |
| CPAR     | PA    | <i>LL_DMA_ConfigAddresses</i>         |
|          |       | <i>LL_DMA_GetM2MSrcAddress</i>        |
|          |       | <i>LL_DMA_GetPeriphAddress</i>        |
|          |       | <i>LL_DMA_SetM2MSrcAddress</i>        |
|          |       | <i>LL_DMA_SetPeriphAddress</i>        |
| IFCR     | CGIF1 | <i>LL_DMA_ClearFlag_GI1</i>           |
|          | CGIF2 | <i>LL_DMA_ClearFlag_GI2</i>           |
|          | CGIF3 | <i>LL_DMA_ClearFlag_GI3</i>           |
|          | CGIF4 | <i>LL_DMA_ClearFlag_GI4</i>           |
|          | CGIF5 | <i>LL_DMA_ClearFlag_GI5</i>           |

| Register | Field  | Function                                       |
|----------|--------|------------------------------------------------|
|          | CGIF6  | <a href="#"><i>LL_DMA_ClearFlag_GI6</i></a>    |
|          | CGIF7  | <a href="#"><i>LL_DMA_ClearFlag_GI7</i></a>    |
|          | CHTIF1 | <a href="#"><i>LL_DMA_ClearFlag_HT1</i></a>    |
|          | CHTIF2 | <a href="#"><i>LL_DMA_ClearFlag_HT2</i></a>    |
|          | CHTIF3 | <a href="#"><i>LL_DMA_ClearFlag_HT3</i></a>    |
|          | CHTIF4 | <a href="#"><i>LL_DMA_ClearFlag_HT4</i></a>    |
|          | CHTIF5 | <a href="#"><i>LL_DMA_ClearFlag_HT5</i></a>    |
|          | CHTIF6 | <a href="#"><i>LL_DMA_ClearFlag_HT6</i></a>    |
|          | CHTIF7 | <a href="#"><i>LL_DMA_ClearFlag_HT7</i></a>    |
|          | CTCIF1 | <a href="#"><i>LL_DMA_ClearFlag_TC1</i></a>    |
|          | CTCIF2 | <a href="#"><i>LL_DMA_ClearFlag_TC2</i></a>    |
|          | CTCIF3 | <a href="#"><i>LL_DMA_ClearFlag_TC3</i></a>    |
|          | CTCIF4 | <a href="#"><i>LL_DMA_ClearFlag_TC4</i></a>    |
|          | CTCIF5 | <a href="#"><i>LL_DMA_ClearFlag_TC5</i></a>    |
|          | CTCIF6 | <a href="#"><i>LL_DMA_ClearFlag_TC6</i></a>    |
|          | CTCIF7 | <a href="#"><i>LL_DMA_ClearFlag_TC7</i></a>    |
|          | CTEIF1 | <a href="#"><i>LL_DMA_ClearFlag_TE1</i></a>    |
|          | CTEIF2 | <a href="#"><i>LL_DMA_ClearFlag_TE2</i></a>    |
|          | CTEIF3 | <a href="#"><i>LL_DMA_ClearFlag_TE3</i></a>    |
|          | CTEIF4 | <a href="#"><i>LL_DMA_ClearFlag_TE4</i></a>    |
|          | CTEIF5 | <a href="#"><i>LL_DMA_ClearFlag_TE5</i></a>    |
|          | CTEIF6 | <a href="#"><i>LL_DMA_ClearFlag_TE6</i></a>    |
|          | CTEIF7 | <a href="#"><i>LL_DMA_ClearFlag_TE7</i></a>    |
| ISR      | GIF1   | <a href="#"><i>LL_DMA_IsActiveFlag_GI1</i></a> |
|          | GIF2   | <a href="#"><i>LL_DMA_IsActiveFlag_GI2</i></a> |
|          | GIF3   | <a href="#"><i>LL_DMA_IsActiveFlag_GI3</i></a> |
|          | GIF4   | <a href="#"><i>LL_DMA_IsActiveFlag_GI4</i></a> |
|          | GIF5   | <a href="#"><i>LL_DMA_IsActiveFlag_GI5</i></a> |
|          | GIF6   | <a href="#"><i>LL_DMA_IsActiveFlag_GI6</i></a> |
|          | GIF7   | <a href="#"><i>LL_DMA_IsActiveFlag_GI7</i></a> |
|          | HTIF1  | <a href="#"><i>LL_DMA_IsActiveFlag_HT1</i></a> |
|          | HTIF2  | <a href="#"><i>LL_DMA_IsActiveFlag_HT2</i></a> |
|          | HTIF3  | <a href="#"><i>LL_DMA_IsActiveFlag_HT3</i></a> |
|          | HTIF4  | <a href="#"><i>LL_DMA_IsActiveFlag_HT4</i></a> |
|          | HTIF5  | <a href="#"><i>LL_DMA_IsActiveFlag_HT5</i></a> |
|          | HTIF6  | <a href="#"><i>LL_DMA_IsActiveFlag_HT6</i></a> |

| Register | Field | Function                                |
|----------|-------|-----------------------------------------|
|          | HTIF7 | <a href="#">LL_DMA_IsActiveFlag_HT7</a> |
|          | TCIF1 | <a href="#">LL_DMA_IsActiveFlag_TC1</a> |
|          | TCIF2 | <a href="#">LL_DMA_IsActiveFlag_TC2</a> |
|          | TCIF3 | <a href="#">LL_DMA_IsActiveFlag_TC3</a> |
|          | TCIF4 | <a href="#">LL_DMA_IsActiveFlag_TC4</a> |
|          | TCIF5 | <a href="#">LL_DMA_IsActiveFlag_TC5</a> |
|          | TCIF6 | <a href="#">LL_DMA_IsActiveFlag_TC6</a> |
|          | TCIF7 | <a href="#">LL_DMA_IsActiveFlag_TC7</a> |
|          | TEIF1 | <a href="#">LL_DMA_IsActiveFlag_TE1</a> |
|          | TEIF2 | <a href="#">LL_DMA_IsActiveFlag_TE2</a> |
|          | TEIF3 | <a href="#">LL_DMA_IsActiveFlag_TE3</a> |
|          | TEIF4 | <a href="#">LL_DMA_IsActiveFlag_TE4</a> |
|          | TEIF5 | <a href="#">LL_DMA_IsActiveFlag_TE5</a> |
|          | TEIF6 | <a href="#">LL_DMA_IsActiveFlag_TE6</a> |
|          | TEIF7 | <a href="#">LL_DMA_IsActiveFlag_TE7</a> |

## 70.8 EXTI

Table 32: Correspondence between EXTI registers and EXTI low-layer driver functions

| Register | Field | Function                                          |
|----------|-------|---------------------------------------------------|
| EMR      | EMx   | <a href="#">LL_EXTI_DisableEvent_0_31</a>         |
|          |       | <a href="#">LL_EXTI_EnableEvent_0_31</a>          |
|          |       | <a href="#">LL_EXTI_IsEnabledEvent_0_31</a>       |
| FTSR     | FTx   | <a href="#">LL_EXTI_DisableFallingTrig_0_31</a>   |
|          |       | <a href="#">LL_EXTI_EnableFallingTrig_0_31</a>    |
|          |       | <a href="#">LL_EXTI_IsEnabledFallingTrig_0_31</a> |
| IMR      | IMx   | <a href="#">LL_EXTI_DisableIT_0_31</a>            |
|          |       | <a href="#">LL_EXTI_EnableIT_0_31</a>             |
|          |       | <a href="#">LL_EXTI_IsEnabledIT_0_31</a>          |
| PR       | PIFx  | <a href="#">LL_EXTI_ClearFlag_0_31</a>            |
|          |       | <a href="#">LL_EXTI_IsActiveFlag_0_31</a>         |
|          |       | <a href="#">LL_EXTI_ReadFlag_0_31</a>             |
| RTSR     | RTx   | <a href="#">LL_EXTI_DisableRisingTrig_0_31</a>    |
|          |       | <a href="#">LL_EXTI_EnableRisingTrig_0_31</a>     |
|          |       | <a href="#">LL_EXTI_IsEnabledRisingTrig_0_31</a>  |
| SWIER    | SWIx  | <a href="#">LL_EXTI_GenerateSWI_0_31</a>          |

## 70.9 GPIO

Table 33: Correspondence between GPIO registers and GPIO low-layer driver functions

| Register | Field   | Function                        |
|----------|---------|---------------------------------|
| AFRH     | AFSELy  | <i>LL_GPIO_GetAFPin_8_15</i>    |
|          |         | <i>LL_GPIO_SetAFPin_8_15</i>    |
| AFRL     | AFSELy  | <i>LL_GPIO_GetAFPin_0_7</i>     |
|          |         | <i>LL_GPIO_SetAFPin_0_7</i>     |
| BRR      | BRy     | <i>LL_GPIO_ResetOutputPin</i>   |
| BSRR     | BRy     | <i>LL_GPIO_ResetOutputPin</i>   |
|          | BSy     | <i>LL_GPIO_SetOutputPin</i>     |
| IDR      | IDy     | <i>LL_GPIO_IsInputPinSet</i>    |
|          |         | <i>LL_GPIO_ReadInputPort</i>    |
| LCKR     | LCKK    | <i>LL_GPIO_IsAnyPinLocked</i>   |
|          |         | <i>LL_GPIO_LockPin</i>          |
| LCKR     | LCKy    | <i>LL_GPIO_IsPinLocked</i>      |
|          |         |                                 |
| MODER    | MODEy   | <i>LL_GPIO_GetPinMode</i>       |
|          |         | <i>LL_GPIO_SetPinMode</i>       |
| ODR      | ODy     | <i>LL_GPIO_IsOutputPinSet</i>   |
|          |         | <i>LL_GPIO_ReadOutputPort</i>   |
|          |         | <i>LL_GPIO_TogglePin</i>        |
|          |         | <i>LL_GPIO_WriteOutputPort</i>  |
| OSPEEDR  | OSPEEDy | <i>LL_GPIO_GetPinSpeed</i>      |
|          |         | <i>LL_GPIO_SetPinSpeed</i>      |
| OTYPER   | OTy     | <i>LL_GPIO_GetPinOutputType</i> |
|          |         | <i>LL_GPIO_SetPinOutputType</i> |
| PUPDR    | PUPDy   | <i>LL_GPIO_GetPinPull</i>       |
|          |         | <i>LL_GPIO_SetPinPull</i>       |

## 70.10 I2C

Table 34: Correspondence between I2C registers and I2C low-layer driver functions

| Register | Field | Function                     |
|----------|-------|------------------------------|
| CCR      | CCR   | <i>LL_I2C_ConfigSpeed</i>    |
|          |       | <i>LL_I2C_GetClockPeriod</i> |
|          |       | <i>LL_I2C_SetClockPeriod</i> |
|          | DUTY  | <i>LL_I2C_ConfigSpeed</i>    |
|          |       | <i>LL_I2C_GetDutyCycle</i>   |
|          |       | <i>LL_I2C_SetDutyCycle</i>   |

| Register | Field     | Function                                |
|----------|-----------|-----------------------------------------|
| CR1      | FS        | <i>LL_I2C_ConfigSpeed</i>               |
|          |           | <i>LL_I2C_GetClockSpeedMode</i>         |
|          |           | <i>LL_I2C_SetClockSpeedMode</i>         |
|          | ACK       | <i>LL_I2C_AcknowledgeNextData</i>       |
|          | ALERT     | <i>LL_I2C_DisableSMBusAlert</i>         |
|          |           | <i>LL_I2C_EnableSMBusAlert</i>          |
|          |           | <i>LL_I2C_IsEnabledSMBusAlert</i>       |
|          | ENARP     | <i>LL_I2C_GetMode</i>                   |
|          |           | <i>LL_I2C_SetMode</i>                   |
|          | ENG C     | <i>LL_I2C_DisableGeneralCall</i>        |
|          |           | <i>LL_I2C_EnableGeneralCall</i>         |
|          |           | <i>LL_I2C_IsEnabledGeneralCall</i>      |
|          | ENPEC     | <i>LL_I2C_DisableSMBusPEC</i>           |
|          |           | <i>LL_I2C_EnableSMBusPEC</i>            |
|          |           | <i>LL_I2C_IsEnabledSMBusPEC</i>         |
|          | NOSTRETCH | <i>LL_I2C_DisableClockStretching</i>    |
|          |           | <i>LL_I2C_EnableClockStretching</i>     |
|          |           | <i>LL_I2C_IsEnabledClockStretching</i>  |
|          | PE        | <i>LL_I2C_ClearFlag_STOP</i>            |
|          |           | <i>LL_I2C_Disable</i>                   |
|          |           | <i>LL_I2C_Enable</i>                    |
|          |           | <i>LL_I2C_IsEnabled</i>                 |
|          | PEC       | <i>LL_I2C_DisableSMBusPECCCompare</i>   |
|          |           | <i>LL_I2C_EnableSMBusPECCCompare</i>    |
|          |           | <i>LL_I2C_IsEnabledSMBusPECCCompare</i> |
|          | POS       | <i>LL_I2C_DisableBitPOS</i>             |
|          |           | <i>LL_I2C_EnableBitPOS</i>              |
|          |           | <i>LL_I2C_IsEnabledBitPOS</i>           |
|          | SMBTYPE   | <i>LL_I2C_GetMode</i>                   |
|          |           | <i>LL_I2C_SetMode</i>                   |
|          | SMBUS     | <i>LL_I2C_GetMode</i>                   |
|          |           | <i>LL_I2C_SetMode</i>                   |
|          | START     | <i>LL_I2C_GenerateStartCondition</i>    |
|          | STOP      | <i>LL_I2C_GenerateStopCondition</i>     |
|          | SWRST     | <i>LL_I2C_DisableReset</i>              |
|          |           | <i>LL_I2C_EnableReset</i>               |

| Register | Field   | Function                         |
|----------|---------|----------------------------------|
|          |         | <i>LL_I2C_IsResetEnabled</i>     |
| CR2      | DMAEN   | <i>LL_I2C_DisableDMAReq_RX</i>   |
|          |         | <i>LL_I2C_DisableDMAReq_TX</i>   |
|          |         | <i>LL_I2C_EnableDMAReq_RX</i>    |
|          |         | <i>LL_I2C_EnableDMAReq_TX</i>    |
|          |         | <i>LL_I2C_IsEnabledDMAReq_RX</i> |
|          |         | <i>LL_I2C_IsEnabledDMAReq_TX</i> |
| FREQ     | FREQ    | <i>LL_I2C_ConfigSpeed</i>        |
|          |         | <i>LL_I2C_GetPeriphClock</i>     |
|          |         | <i>LL_I2C_SetPeriphClock</i>     |
|          | ITBUFEN | <i>LL_I2C_DisableIT_BUF</i>      |
|          |         | <i>LL_I2C_DisableIT_RX</i>       |
|          |         | <i>LL_I2C_DisableIT_TX</i>       |
| ITERREN  | ITERREN | <i>LL_I2C_EnableIT_BUF</i>       |
|          |         | <i>LL_I2C_EnableIT_RX</i>        |
|          |         | <i>LL_I2C_EnableIT_TX</i>        |
|          | ITERREN | <i>LL_I2C_IsEnabledIT_BUF</i>    |
|          |         | <i>LL_I2C_IsEnabledIT_RX</i>     |
|          |         | <i>LL_I2C_IsEnabledIT_TX</i>     |
| ITEVTEN  | ITEVTEN | <i>LL_I2C_DisableIT_ERR</i>      |
|          |         | <i>LL_I2C_EnableIT_ERR</i>       |
|          |         | <i>LL_I2C_IsEnabledIT_ERR</i>    |
|          |         | <i>LL_I2C_DisableIT_EVT</i>      |
|          |         | <i>LL_I2C_DisableIT_RX</i>       |
|          |         | <i>LL_I2C_DisableIT_TX</i>       |
| LAST     | LAST    | <i>LL_I2C_EnableIT_EVT</i>       |
|          |         | <i>LL_I2C_EnableIT_RX</i>        |
|          |         | <i>LL_I2C_EnableIT_TX</i>        |
|          | LAST    | <i>LL_I2C_IsEnabledIT_EVT</i>    |
|          |         | <i>LL_I2C_IsEnabledIT_RX</i>     |
|          |         | <i>LL_I2C_IsEnabledIT_TX</i>     |
| DR       | DR      | <i>LL_I2C_DisableLastDMA</i>     |
|          |         | <i>LL_I2C_EnableLastDMA</i>      |
| DR       | DR      | <i>LL_I2C_IsEnabledLastDMA</i>   |
|          |         | <i>LL_I2C_DMA_GetRegAddr</i>     |
|          |         | <i>LL_I2C_ReceiveData8</i>       |

| Register | Field    | Function                                |
|----------|----------|-----------------------------------------|
| OAR1     | ADD0     | <i>LL_I2C_TransmitData8</i>             |
|          | ADD1_7   | <i>LL_I2C_SetOwnAddress1</i>            |
|          | ADD8_9   | <i>LL_I2C_SetOwnAddress1</i>            |
|          | ADDMODE  | <i>LL_I2C_SetOwnAddress1</i>            |
| OAR2     | ADD2     | <i>LL_I2C_SetOwnAddress2</i>            |
|          | ENDUAL   | <i>LL_I2C_DisableOwnAddress2</i>        |
|          |          | <i>LL_I2C_EnableOwnAddress2</i>         |
|          |          | <i>LL_I2C_IsEnabledOwnAddress2</i>      |
| SR1      | ADD10    | <i>LL_I2C_IsActiveFlag_ADD10</i>        |
|          | ADDR     | <i>LL_I2C_ClearFlag_ADDR</i>            |
|          |          | <i>LL_I2C_IsActiveFlag_ADDR</i>         |
|          | AF       | <i>LL_I2C_ClearFlag_AF</i>              |
|          |          | <i>LL_I2C_IsActiveFlag_AF</i>           |
|          | ARLO     | <i>LL_I2C_ClearFlag_ARLO</i>            |
|          |          | <i>LL_I2C_IsActiveFlag_ARLO</i>         |
|          | BERR     | <i>LL_I2C_ClearFlag_BERR</i>            |
|          |          | <i>LL_I2C_IsActiveFlag_BERR</i>         |
|          | BTF      | <i>LL_I2C_IsActiveFlag_BTF</i>          |
|          | OVR      | <i>LL_I2C_ClearFlag_OVR</i>             |
|          |          | <i>LL_I2C_IsActiveFlag_OVR</i>          |
|          | PECERR   | <i>LL_I2C_ClearSMBusFlag_PECERR</i>     |
|          |          | <i>LL_I2C_IsActiveSMBusFlag_PECERR</i>  |
|          | RXNE     | <i>LL_I2C_IsActiveFlag_RXNE</i>         |
|          | SB       | <i>LL_I2C_IsActiveFlag_SB</i>           |
|          | SMBALERT | <i>LL_I2C_ClearSMBusFlag_ALERT</i>      |
|          |          | <i>LL_I2C_IsActiveSMBusFlag_ALERT</i>   |
|          | STOPF    | <i>LL_I2C_ClearFlag_STOP</i>            |
|          |          | <i>LL_I2C_IsActiveFlag_STOP</i>         |
|          | TIMEOUT  | <i>LL_I2C_ClearSMBusFlag_TIMEOUT</i>    |
|          |          | <i>LL_I2C_IsActiveSMBusFlag_TIMEOUT</i> |
|          | TXE      | <i>LL_I2C_IsActiveFlag_TXE</i>          |
| SR2      | BUSY     | <i>LL_I2C_IsActiveFlag_BUSY</i>         |
|          | DUALF    | <i>LL_I2C_IsActiveFlag_DUAL</i>         |
|          | GENCALL  | <i>LL_I2C_IsActiveFlag_GENCALL</i>      |
|          | MSL      | <i>LL_I2C_IsActiveFlag_MSL</i>          |

| Register | Field      | Function                                   |
|----------|------------|--------------------------------------------|
|          | PEC        | <i>LL_I2C_GetSMBusPEC</i>                  |
|          | SMBDEFAULT | <i>LL_I2C_IsActiveSMBusFlag_SMBDEFAULT</i> |
|          | SMBHOST    | <i>LL_I2C_IsActiveSMBusFlag_SMBHOST</i>    |
|          | TRA        | <i>LL_I2C_GetTransferDirection</i>         |
| TRISE    | TRISE      | <i>LL_I2C_ConfigSpeed</i>                  |
|          |            | <i>LL_I2C_GetRiseTime</i>                  |
|          |            | <i>LL_I2C_SetRiseTime</i>                  |

## 70.11 I2S

Table 35: Correspondence between I2S registers and I2S low-layer driver functions

| Register | Field   | Function                         |
|----------|---------|----------------------------------|
| CR2      | ERRIE   | <i>LL_I2S_DisableIT_ERR</i>      |
|          |         | <i>LL_I2S_EnableIT_ERR</i>       |
|          |         | <i>LL_I2S_IsEnabledIT_ERR</i>    |
|          | RXDMAEN | <i>LL_I2S_DisableDMAReq_RX</i>   |
|          |         | <i>LL_I2S_EnableDMAReq_RX</i>    |
|          |         | <i>LL_I2S_IsEnabledDMAReq_RX</i> |
|          | RXNEIE  | <i>LL_I2S_DisableIT_RXNE</i>     |
|          |         | <i>LL_I2S_EnableIT_RXNE</i>      |
|          |         | <i>LL_I2S_IsEnabledIT_RXNE</i>   |
|          | TXDMAEN | <i>LL_I2S_DisableDMAReq_TX</i>   |
|          |         | <i>LL_I2S_EnableDMAReq_TX</i>    |
|          |         | <i>LL_I2S_IsEnabledDMAReq_TX</i> |
|          | TXEIE   | <i>LL_I2S_DisableIT_TXE</i>      |
|          |         | <i>LL_I2S_EnableIT_TXE</i>       |
|          |         | <i>LL_I2S_IsEnabledIT_TXE</i>    |
| DR       | DR      | <i>LL_I2S_ReceiveData16</i>      |
|          |         | <i>LL_I2S_TransmitData16</i>     |
| I2SCFGR  | CHLEN   | <i>LL_I2S_GetDataFormat</i>      |
|          |         | <i>LL_I2S_SetDataFormat</i>      |
|          | CKPOL   | <i>LL_I2S_GetClockPolarity</i>   |
|          |         | <i>LL_I2S_SetClockPolarity</i>   |
|          | DATLEN  | <i>LL_I2S_GetDataFormat</i>      |
|          |         | <i>LL_I2S_SetDataFormat</i>      |
|          | I2SCFG  | <i>LL_I2S_GetTransferMode</i>    |
|          |         | <i>LL_I2S_SetTransferMode</i>    |

| Register | Field   | Function                           |
|----------|---------|------------------------------------|
| I2S      | I2SE    | <i>LL_I2S_Disable</i>              |
|          |         | <i>LL_I2S_Enable</i>               |
|          |         | <i>LL_I2S_IsEnabled</i>            |
|          | I2SMOD  | <i>LL_I2S_Enable</i>               |
|          | I2STD   | <i>LL_I2S_GetStandard</i>          |
|          |         | <i>LL_I2S_SetStandard</i>          |
|          | PCMSYNC | <i>LL_I2S_GetStandard</i>          |
|          |         | <i>LL_I2S_SetStandard</i>          |
|          | I2SDIV  | <i>LL_I2S_GetPrescalerLinear</i>   |
|          |         | <i>LL_I2S_SetPrescalerLinear</i>   |
| I2SPR    | MCKOE   | <i>LL_I2S_DisableMasterClock</i>   |
|          |         | <i>LL_I2S_EnableMasterClock</i>    |
|          |         | <i>LL_I2S_IsEnabledMasterClock</i> |
|          | ODD     | <i>LL_I2S_GetPrescalerParity</i>   |
|          |         | <i>LL_I2S_SetPrescalerParity</i>   |
| SR       | BSY     | <i>LL_I2S_IsActiveFlag_BSY</i>     |
|          | CHSIDE  | <i>LL_I2S_IsActiveFlag_CHSIDE</i>  |
|          | FRE     | <i>LL_I2S_ClearFlag_FRE</i>        |
|          |         | <i>LL_I2S_IsActiveFlag_FRE</i>     |
|          | OVR     | <i>LL_I2S_ClearFlag_OVR</i>        |
|          |         | <i>LL_I2S_IsActiveFlag_OVR</i>     |
|          | RXNE    | <i>LL_I2S_IsActiveFlag_RXNE</i>    |
|          | TXE     | <i>LL_I2S_IsActiveFlag_TXE</i>     |
|          | UDR     | <i>LL_I2S_ClearFlag_UDR</i>        |
|          |         | <i>LL_I2S_IsActiveFlag_UDR</i>     |

## 70.12 IWDG

Table 36: Correspondence between IWDG registers and IWDG low-layer driver functions

| Register | Field | Function                          |
|----------|-------|-----------------------------------|
| KR       | KEY   | <i>LL_IWDG_DisableWriteAccess</i> |
|          |       | <i>LL_IWDG_Enable</i>             |
|          |       | <i>LL_IWDG_EnableWriteAccess</i>  |
|          |       | <i>LL_IWDG_ReloadCounter</i>      |
| PR       | PR    | <i>LL_IWDG_GetPrescaler</i>       |
|          |       | <i>LL_IWDG_SetPrescaler</i>       |
| RLR      | RL    | <i>LL_IWDG_GetReloadCounter</i>   |

| Register | Field | Function                                        |
|----------|-------|-------------------------------------------------|
| SR       | PVU   | <a href="#"><i>LL_IWDG_SetReloadCounter</i></a> |
|          |       | <a href="#"><i>LL_IWDG_IsActiveFlag_PVU</i></a> |
|          |       | <a href="#"><i>LL_IWDG_IsReady</i></a>          |
|          | RVU   | <a href="#"><i>LL_IWDG_IsActiveFlag_RVU</i></a> |
|          |       | <a href="#"><i>LL_IWDG_IsReady</i></a>          |

## 70.13 OPAMP

Table 37: Correspondence between OPAMP registers and OPAMP low-layer driver functions

| Register | Field      | Function                                                |
|----------|------------|---------------------------------------------------------|
| CSR      | ANAWSELx   | <a href="#"><i>LL_OPAMP_SetInputInverting</i></a>       |
|          | AOP_RANGE  | <a href="#"><i>LL_OPAMP_GetCommonPowerRange</i></a>     |
|          |            | <a href="#"><i>LL_OPAMP_SetCommonPowerRange</i></a>     |
|          | OPA1CAL_H  | <a href="#"><i>LL_OPAMP_SetCalibrationSelection</i></a> |
|          | OPA1CAL_L  | <a href="#"><i>LL_OPAMP_SetCalibrationSelection</i></a> |
|          | OPA1LPM    | <a href="#"><i>LL_OPAMP_GetPowerMode</i></a>            |
|          |            | <a href="#"><i>LL_OPAMP_SetPowerMode</i></a>            |
|          | OPA2LPM    | <a href="#"><i>LL_OPAMP_GetPowerMode</i></a>            |
|          |            | <a href="#"><i>LL_OPAMP_SetPowerMode</i></a>            |
|          | OPA3LPM    | <a href="#"><i>LL_OPAMP_GetPowerMode</i></a>            |
|          |            | <a href="#"><i>LL_OPAMP_SetPowerMode</i></a>            |
|          | OPAxCALOUT | <a href="#"><i>LL_OPAMP_IsCalibrationOutputSet</i></a>  |
|          | OPAxPD     | <a href="#"><i>LL_OPAMP_Disable</i></a>                 |
|          |            | <a href="#"><i>LL_OPAMP_Enable</i></a>                  |
|          |            | <a href="#"><i>LL_OPAMP_IsEnabled</i></a>               |
|          | S3SELx     | <a href="#"><i>LL_OPAMP_GetFunctionalMode</i></a>       |
|          |            | <a href="#"><i>LL_OPAMP_SetFunctionalMode</i></a>       |
|          |            | <a href="#"><i>LL_OPAMP_SetMode</i></a>                 |
|          | S4SELx     | <a href="#"><i>LL_OPAMP_SetInputInverting</i></a>       |
|          |            | <a href="#"><i>LL_OPAMP_SetMode</i></a>                 |
|          | S5SELx     | <a href="#"><i>LL_OPAMP_GetInputNonInverting</i></a>    |
|          |            | <a href="#"><i>LL_OPAMP_SetInputNonInverting</i></a>    |
|          |            | <a href="#"><i>LL_OPAMP_SetMode</i></a>                 |
|          | S6SELx     | <a href="#"><i>LL_OPAMP_GetInputNonInverting</i></a>    |
|          |            | <a href="#"><i>LL_OPAMP_SetInputNonInverting</i></a>    |
|          |            | <a href="#"><i>LL_OPAMP_SetMode</i></a>                 |
|          | S7SEL2     | <a href="#"><i>LL_OPAMP_GetInputNonInverting</i></a>    |

| Register | Field                       | Function                              |
|----------|-----------------------------|---------------------------------------|
|          |                             | <i>LL_OPAMP_SetInputNonInverting</i>  |
|          |                             | <i>LL_OPAMP_SetMode</i>               |
| LPOTR    | AOx_OPT_OFFSET_TRIM_LP_HIGH | <i>LL_OPAMP_GetTrimmingValue</i>      |
|          |                             | <i>LL_OPAMP_SetTrimmingValue</i>      |
| OTR      | AOx_OPT_OFFSET_TRIM_LP_LOW  | <i>LL_OPAMP_GetTrimmingValue</i>      |
|          |                             | <i>LL_OPAMP_SetTrimmingValue</i>      |
|          | AOx_OPT_OFFSET_TRIM_HIGH    | <i>LL_OPAMP_GetTrimmingValue</i>      |
|          |                             | <i>LL_OPAMP_SetTrimmingValue</i>      |
|          | AOx_OPT_OFFSET_TRIM_LOW     | <i>LL_OPAMP_GetTrimmingValue</i>      |
|          |                             | <i>LL_OPAMP_SetTrimmingValue</i>      |
|          | OT_USER                     | <i>LL_OPAMP_GetCommonTrimmingMode</i> |
|          |                             | <i>LL_OPAMP_SetCommonTrimmingMode</i> |

## 70.14 PWR

Table 38: Correspondence between PWR registers and PWR low-layer driver functions

| Register | Field  | Function                               |
|----------|--------|----------------------------------------|
| CR       | CSBF   | <i>LL_PWR_ClearFlag_SB</i>             |
|          | CWUF   | <i>LL_PWR_ClearFlag_WU</i>             |
|          | DBP    | <i>LL_PWR_DisableBkUpAccess</i>        |
|          |        | <i>LL_PWR_EnableBkUpAccess</i>         |
|          |        | <i>LL_PWR_IsEnabledBkUpAccess</i>      |
|          | FWU    | <i>LL_PWR_DisableFastWakeUp</i>        |
|          |        | <i>LL_PWR_EnableFastWakeUp</i>         |
|          |        | <i>LL_PWR_IsEnabledFastWakeUp</i>      |
|          | LPRUN  | <i>LL_PWR_DisableLowPowerRunMode</i>   |
|          |        | <i>LL_PWR_EnableLowPowerRunMode</i>    |
|          |        | <i>LL_PWR_EnterLowPowerRunMode</i>     |
|          |        | <i>LL_PWR_ExitLowPowerRunMode</i>      |
|          |        | <i>LL_PWR_IsEnabledLowPowerRunMode</i> |
|          | LPSDSR | <i>LL_PWR_EnterLowPowerRunMode</i>     |
|          |        | <i>LL_PWR_ExitLowPowerRunMode</i>      |
|          |        | <i>LL_PWR_GetRegullModeLP</i>          |
|          |        | <i>LL_PWR_SetRegullModeLP</i>          |
|          | PDDS   | <i>LL_PWR_GetPowerMode</i>             |
|          |        | <i>LL_PWR_SetPowerMode</i>             |
|          | PLS    | <i>LL_PWR_GetPVDLevel</i>              |

| Register | Field       | Function                              |
|----------|-------------|---------------------------------------|
| CSR      | PVDE        | <i>LL_PWR_SetPVDLevel</i>             |
|          |             | <i>LL_PWR_DisablePVD</i>              |
|          |             | <i>LL_PWR_EnablePVD</i>               |
|          |             | <i>LL_PWR_IsEnabledPVD</i>            |
|          | ULP         | <i>LL_PWR_DisableUltraLowPower</i>    |
|          |             | <i>LL_PWR_EnableUltraLowPower</i>     |
|          |             | <i>LL_PWR_IsEnabledUltraLowPower</i>  |
|          | VOS         | <i>LL_PWR_GetRegulVoltageScaling</i>  |
|          |             | <i>LL_PWR_SetRegulVoltageScaling</i>  |
|          | EWUP1       | <i>LL_PWR_DisableWakeUpPin</i>        |
|          |             | <i>LL_PWR_EnableWakeUpPin</i>         |
|          |             | <i>LL_PWR_IsEnabledWakeUpPin</i>      |
|          | EWUP2       | <i>LL_PWR_DisableWakeUpPin</i>        |
|          |             | <i>LL_PWR_EnableWakeUpPin</i>         |
|          |             | <i>LL_PWR_IsEnabledWakeUpPin</i>      |
|          | EWUP3       | <i>LL_PWR_DisableWakeUpPin</i>        |
|          |             | <i>LL_PWR_EnableWakeUpPin</i>         |
|          |             | <i>LL_PWR_IsEnabledWakeUpPin</i>      |
|          | PVDO        | <i>LL_PWR_IsActiveFlag_PVDO</i>       |
|          | REGLPF      | <i>LL_PWR_IsActiveFlag_REGLPF</i>     |
|          | SBF         | <i>LL_PWR_IsActiveFlag_SB</i>         |
|          | VOSF        | <i>LL_PWR_IsActiveFlag_VOS</i>        |
|          | VREFINTRDYF | <i>LL_PWR_IsActiveFlag_VREFINTRDY</i> |
|          | WUF         | <i>LL_PWR_IsActiveFlag_WU</i>         |

## 70.15 RCC

Table 39: Correspondence between RCC registers and RCC low-layer driver functions

| Register | Field  | Function                           |
|----------|--------|------------------------------------|
| CFGR     | HPRE   | <i>LL_RCC_GetAHBPrescaler</i>      |
|          |        | <i>LL_RCC_SetAHBPrescaler</i>      |
|          | MCPRE  | <i>LL_RCC_ConfigMCO</i>            |
|          | MCOSEL | <i>LL_RCC_ConfigMCO</i>            |
|          | PLLDIV | <i>LL_RCC_PLL_ConfigDomain_SYS</i> |
|          |        | <i>LL_RCC_PLL_GetDivider</i>       |
|          | PLLMUL | <i>LL_RCC_PLL_ConfigDomain_SYS</i> |
|          |        | <i>LL_RCC_PLL_GetMultiplicator</i> |

| Register | Field    | Function                           |
|----------|----------|------------------------------------|
| CIR      | PLLSRC   | <i>LL_RCC_PLL_ConfigDomain_SYS</i> |
|          |          | <i>LL_RCC_PLL_GetMainSource</i>    |
|          | PPRE1    | <i>LL_RCC_GetAPB1Prescaler</i>     |
|          |          | <i>LL_RCC_SetAPB1Prescaler</i>     |
|          | PPRE2    | <i>LL_RCC_GetAPB2Prescaler</i>     |
|          |          | <i>LL_RCC_SetAPB2Prescaler</i>     |
|          | SW       | <i>LL_RCC_SetSysClkSource</i>      |
|          | SWS      | <i>LL_RCC_GetSysClkSource</i>      |
|          | CSSC     | <i>LL_RCC_ClearFlag_HSECSS</i>     |
|          | CSSF     | <i>LL_RCC_IsActiveFlag_HSECSS</i>  |
|          | HSERDYC  | <i>LL_RCC_ClearFlag_HSERDY</i>     |
|          | HSERDYF  | <i>LL_RCC_IsActiveFlag_HSERDY</i>  |
|          | HSERDYIE | <i>LL_RCC_DisableIT_HSERDY</i>     |
|          |          | <i>LL_RCC_EnableIT_HSERDY</i>      |
|          |          | <i>LL_RCC_IsEnabledIT_HSERDY</i>   |
|          | HSIRDYC  | <i>LL_RCC_ClearFlag_HSIRDY</i>     |
|          | HSIRDYF  | <i>LL_RCC_IsActiveFlag_HSIRDY</i>  |
|          | HSIRDYIE | <i>LL_RCC_DisableIT_HSIRDY</i>     |
|          |          | <i>LL_RCC_EnableIT_HSIRDY</i>      |
|          |          | <i>LL_RCC_IsEnabledIT_HSIRDY</i>   |
|          | LSECSSC  | <i>LL_RCC_ClearFlag_LSECSS</i>     |
|          | LSECSSF  | <i>LL_RCC_IsActiveFlag_LSECSS</i>  |
|          | LSECSSIE | <i>LL_RCC_DisableIT_LSECSS</i>     |
|          |          | <i>LL_RCC_EnableIT_LSECSS</i>      |
|          |          | <i>LL_RCC_IsEnabledIT_LSECSS</i>   |
|          | LSERDYC  | <i>LL_RCC_ClearFlag_LSERDY</i>     |
|          | LSERDYF  | <i>LL_RCC_IsActiveFlag_LSERDY</i>  |
|          | LSERDYIE | <i>LL_RCC_DisableIT_LSERDY</i>     |
|          |          | <i>LL_RCC_EnableIT_LSERDY</i>      |
|          |          | <i>LL_RCC_IsEnabledIT_LSERDY</i>   |
|          | LSIRDYC  | <i>LL_RCC_ClearFlag_LSIRDY</i>     |
|          | LSIRDYF  | <i>LL_RCC_IsActiveFlag_LSIRDY</i>  |
|          | LSIRDYIE | <i>LL_RCC_DisableIT_LSIRDY</i>     |
|          |          | <i>LL_RCC_EnableIT_LSIRDY</i>      |
|          |          | <i>LL_RCC_IsEnabledIT_LSIRDY</i>   |
|          | MSIRDYC  | <i>LL_RCC_ClearFlag_MSIRDY</i>     |

| Register | Field    | Function                           |
|----------|----------|------------------------------------|
| CR       | MSIRDYF  | <i>LL_RCC_IsActiveFlag_MSIRDY</i>  |
|          | MSIRDYIE | <i>LL_RCC_DisableIT_MSIRDY</i>     |
|          |          | <i>LL_RCC_EnableIT_MSIRDY</i>      |
|          |          | <i>LL_RCC_IsEnabledIT_MSIRDY</i>   |
|          | PLLRDYC  | <i>LL_RCC_ClearFlag_PLLRDY</i>     |
|          | PLLRDYF  | <i>LL_RCC_IsActiveFlag_PLLRDY</i>  |
|          | PLLRDYIE | <i>LL_RCC_DisableIT_PLLRDY</i>     |
|          |          | <i>LL_RCC_EnableIT_PLLRDY</i>      |
|          |          | <i>LL_RCC_IsEnabledIT_PLLRDY</i>   |
|          | CSSON    | <i>LL_RCC_HSE_DisableCSS</i>       |
|          |          | <i>LL_RCC_HSE_EnableCSS</i>        |
|          | HSEBYP   | <i>LL_RCC_HSE_DisableBypass</i>    |
|          |          | <i>LL_RCC_HSE_EnableBypass</i>     |
|          | HSEON    | <i>LL_RCC_HSE_Disable</i>          |
|          |          | <i>LL_RCC_HSE_Enable</i>           |
|          | HSERDY   | <i>LL_RCC_HSE_IsReady</i>          |
|          | HSION    | <i>LL_RCC_HSI_Disable</i>          |
|          |          | <i>LL_RCC_HSI_Enable</i>           |
|          | HSIRDY   | <i>LL_RCC_HSI_IsReady</i>          |
|          | MSION    | <i>LL_RCC_MSI_Disable</i>          |
|          |          | <i>LL_RCC_MSI_Enable</i>           |
|          | MSIRDY   | <i>LL_RCC_MSI_IsReady</i>          |
|          | PLLON    | <i>LL_RCC_PLL_Disable</i>          |
|          |          | <i>LL_RCC_PLL_Enable</i>           |
|          | PLLRDY   | <i>LL_RCC_PLL_IsReady</i>          |
|          | RTCPRE   | <i>LL_RCC_GetRTC_HSEPrescaler</i>  |
|          |          | <i>LL_RCC_SetRTC_HSEPrescaler</i>  |
| CSR      | IWDGRSTF | <i>LL_RCC_IsActiveFlag_IWDGRST</i> |
|          | LPWRRSTF | <i>LL_RCC_IsActiveFlag_LPWRRST</i> |
|          | LSEBYP   | <i>LL_RCC_LSE_DisableBypass</i>    |
|          |          | <i>LL_RCC_LSE_EnableBypass</i>     |
|          | LSECSSD  | <i>LL_RCC_LSE_IsCSSDetected</i>    |
|          | LSECSSON | <i>LL_RCC_LSE_DisableCSS</i>       |
|          |          | <i>LL_RCC_LSE_EnableCSS</i>        |
|          | LSEON    | <i>LL_RCC_LSE_Disable</i>          |
|          |          | <i>LL_RCC_LSE_Enable</i>           |

| Register | Field    | Function                               |
|----------|----------|----------------------------------------|
| ICSCR    | LSERDY   | <i>LL_RCC_LSE_IsReady</i>              |
|          | LSION    | <i>LL_RCC_LSI_Disable</i>              |
|          |          | <i>LL_RCC_LSI_Enable</i>               |
|          | LSIRDY   | <i>LL_RCC_LSI_IsReady</i>              |
|          | OBLRSTF  | <i>LL_RCC_IsActiveFlag_OBLRST</i>      |
|          | PINRSTF  | <i>LL_RCC_IsActiveFlag_PINRST</i>      |
|          | PORRSTF  | <i>LL_RCC_IsActiveFlag_PORRST</i>      |
|          | RMVF     | <i>LL_RCC_ClearResetFlags</i>          |
|          | RTCEN    | <i>LL_RCC_DisableRTC</i>               |
|          |          | <i>LL_RCC_EnableRTC</i>                |
|          |          | <i>LL_RCC_IsEnabledRTC</i>             |
|          | RTCRST   | <i>LL_RCC_ForceBackupDomainReset</i>   |
|          |          | <i>LL_RCC_ReleaseBackupDomainReset</i> |
|          | RTCSEL   | <i>LL_RCC_GetRTCClockSource</i>        |
|          |          | <i>LL_RCC_SetRTCClockSource</i>        |
|          | SFTRSTF  | <i>LL_RCC_IsActiveFlag_SFTRST</i>      |
|          | WWDRGSTF | <i>LL_RCC_IsActiveFlag_WWDGRST</i>     |
|          | ICSCR    | <i>LL_RCC_HSI_GetCalibration</i>       |
|          |          | <i>LL_RCC_HSI_GetCalibTrimming</i>     |
|          |          | <i>LL_RCC_HSI_SetCalibTrimming</i>     |
|          |          | <i>LL_RCC_MSI_GetCalibration</i>       |
|          | MSIRANGE | <i>LL_RCC_MSI_GetRange</i>             |
|          |          | <i>LL_RCC_MSI_SetRange</i>             |
|          | MSITRIM  | <i>LL_RCC_MSI_SetCalibTrimming</i>     |
|          |          | <i>LL_RCC_MSI_SetCalibTrimming</i>     |

## 70.16 RTC

Table 40: Correspondence between RTC registers and RTC low-layer driver functions

| Register | Field | Function                      |
|----------|-------|-------------------------------|
| ALRMAR   | DT    | <i>LL_RTC_ALMA_GetDay</i>     |
|          |       | <i>LL_RTC_ALMA_SetDay</i>     |
|          | DU    | <i>LL_RTC_ALMA_GetDay</i>     |
|          |       | <i>LL_RTC_ALMA_GetWeekDay</i> |
|          |       | <i>LL_RTC_ALMA_SetDay</i>     |
|          |       | <i>LL_RTC_ALMA_SetWeekDay</i> |
|          | HT    | <i>LL_RTC_ALMA_ConfigTime</i> |

| Register | Field | Function                          |
|----------|-------|-----------------------------------|
|          | HU    | <i>LL_RTC_ALMA_GetHour</i>        |
|          |       | <i>LL_RTC_ALMA_SetHour</i>        |
|          |       | <i>LL_RTC_ALMA_ConfigTime</i>     |
|          | MNT   | <i>LL_RTC_ALMA_GetHour</i>        |
|          |       | <i>LL_RTC_ALMA_SetHour</i>        |
|          |       | <i>LL_RTC_ALMA_ConfigTime</i>     |
|          |       | <i>LL_RTC_ALMA_SetMinute</i>      |
|          | MNU   | <i>LL_RTC_ALMA_GetMinute</i>      |
|          |       | <i>LL_RTC_ALMA_SetMinute</i>      |
|          |       | <i>LL_RTC_ALMA_ConfigTime</i>     |
|          |       | <i>LL_RTC_ALMA_GetTime</i>        |
|          | MSK1  | <i>LL_RTC_ALMA_SetTime</i>        |
|          |       | <i>LL_RTC_ALMA_SetMask</i>        |
|          | MSK2  | <i>LL_RTC_ALMA_SetMask</i>        |
|          |       | <i>LL_RTC_ALMA_SetTime</i>        |
|          | MSK3  | <i>LL_RTC_ALMA_SetTime</i>        |
|          |       | <i>LL_RTC_ALMA_SetMask</i>        |
|          | MSK4  | <i>LL_RTC_ALMA_SetTime</i>        |
|          |       | <i>LL_RTC_ALMA_SetMask</i>        |
|          | PM    | <i>LL_RTC_ALMA_ConfigTime</i>     |
|          |       | <i>LL_RTC_ALMA_SetTimeFormat</i>  |
|          |       | <i>LL_RTC_ALMA_GetTimeFormat</i>  |
|          | ST    | <i>LL_RTC_ALMA_SetTimeFormat</i>  |
|          |       | <i>LL_RTC_ALMA_ConfigTime</i>     |
|          |       | <i>LL_RTC_ALMA_SetSecond</i>      |
|          |       | <i>LL_RTC_ALMA_GetTime</i>        |
|          | SU    | <i>LL_RTC_ALMA_SetSecond</i>      |
|          |       | <i>LL_RTC_ALMA_ConfigTime</i>     |
|          |       | <i>LL_RTC_ALMA_SetSecond</i>      |
|          |       | <i>LL_RTC_ALMA_GetTime</i>        |
|          | WDSEL | <i>LL_RTC_ALMA_DisableWeekday</i> |
|          |       | <i>LL_RTC_ALMA_EnableWeekday</i>  |

| Register | Field  | Function                                            |
|----------|--------|-----------------------------------------------------|
| ALRMASSR | MASKSS | <a href="#"><i>LL_RTC_ALMA_GetSubSecondMask</i></a> |
|          |        | <a href="#"><i>LL_RTC_ALMA_SetSubSecondMask</i></a> |
|          | SS     | <a href="#"><i>LL_RTC_ALMA_GetSubSecond</i></a>     |
|          |        | <a href="#"><i>LL_RTC_ALMA_SetSubSecond</i></a>     |
|          | DT     | <a href="#"><i>LL_RTC_ALMB_GetDay</i></a>           |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetDay</i></a>           |
|          | DU     | <a href="#"><i>LL_RTC_ALMB_GetDay</i></a>           |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetWeekDay</i></a>       |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetDay</i></a>           |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetWeekDay</i></a>       |
| ALRMBR   | HT     | <a href="#"><i>LL_RTC_ALMB_ConfigTime</i></a>       |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetHour</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetTime</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetHour</i></a>          |
|          | HU     | <a href="#"><i>LL_RTC_ALMB_ConfigTime</i></a>       |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetHour</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetTime</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetHour</i></a>          |
|          | MNT    | <a href="#"><i>LL_RTC_ALMB_ConfigTime</i></a>       |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetMinute</i></a>        |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetTime</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetMinute</i></a>        |
|          | MNU    | <a href="#"><i>LL_RTC_ALMB_ConfigTime</i></a>       |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetMinute</i></a>        |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetTime</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetMinute</i></a>        |
|          | MSK1   | <a href="#"><i>LL_RTC_ALMB_GetMask</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetMask</i></a>          |
|          | MSK2   | <a href="#"><i>LL_RTC_ALMB_GetMask</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetMask</i></a>          |
|          | MSK3   | <a href="#"><i>LL_RTC_ALMB_GetMask</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetMask</i></a>          |
|          | MSK4   | <a href="#"><i>LL_RTC_ALMB_GetMask</i></a>          |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetMask</i></a>          |
|          | PM     | <a href="#"><i>LL_RTC_ALMB_ConfigTime</i></a>       |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetTimeFormat</i></a>    |

| Register | Field  | Function                                                |
|----------|--------|---------------------------------------------------------|
| ALRMBSSR | ST     | <a href="#"><i>LL_RTC_ALMB_SetTimeFormat</i></a>        |
|          |        | <a href="#"><i>LL_RTC_ALMB_ConfigTime</i></a>           |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetSecond</i></a>            |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetTime</i></a>              |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetSecond</i></a>            |
|          | SU     | <a href="#"><i>LL_RTC_ALMB_ConfigTime</i></a>           |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetSecond</i></a>            |
|          |        | <a href="#"><i>LL_RTC_ALMB_GetTime</i></a>              |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetSecond</i></a>            |
|          | WDSEL  | <a href="#"><i>LL_RTC_ALMB_DisableWeekday</i></a>       |
|          |        | <a href="#"><i>LL_RTC_ALMB_EnableWeekday</i></a>        |
| ALRMBSSR | MASKSS | <a href="#"><i>LL_RTC_ALMB_GetSubSecondMask</i></a>     |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetSubSecondMask</i></a>     |
|          | SS     | <a href="#"><i>LL_RTC_ALMB_GetSubSecond</i></a>         |
|          |        | <a href="#"><i>LL_RTC_ALMB_SetSubSecond</i></a>         |
| BKPxR    | BKP    | <a href="#"><i>LL_RTC_BAK_GetRegister</i></a>           |
|          |        | <a href="#"><i>LL_RTC_BAK_SetRegister</i></a>           |
| CALIBR   | DC     | <a href="#"><i>LL_RTC_CAL_ConfigCoarseDigital</i></a>   |
|          |        | <a href="#"><i>LL_RTC_CAL_GetCoarseDigitalValue</i></a> |
|          | DCS    | <a href="#"><i>LL_RTC_CAL_ConfigCoarseDigital</i></a>   |
|          |        | <a href="#"><i>LL_RTC_CAL_GetCoarseDigitalSign</i></a>  |
| CALR     | CALM   | <a href="#"><i>LL_RTC_CAL_GetMinus</i></a>              |
|          |        | <a href="#"><i>LL_RTC_CAL_SetMinus</i></a>              |
|          | CALP   | <a href="#"><i>LL_RTC_CAL_IsPulseInserted</i></a>       |
|          |        | <a href="#"><i>LL_RTC_CAL_SetPulse</i></a>              |
|          | CALW16 | <a href="#"><i>LL_RTC_CAL_GetPeriod</i></a>             |
|          |        | <a href="#"><i>LL_RTC_CAL_SetPeriod</i></a>             |
|          | CALW8  | <a href="#"><i>LL_RTC_CAL_GetPeriod</i></a>             |
|          |        | <a href="#"><i>LL_RTC_CAL_SetPeriod</i></a>             |
| CR       | ADD1H  | <a href="#"><i>LL_RTC_TIME_IncHour</i></a>              |
|          | ALRAE  | <a href="#"><i>LL_RTC_ALMA_Disable</i></a>              |
|          |        | <a href="#"><i>LL_RTC_ALMA_Enable</i></a>               |
|          | ALRAIE | <a href="#"><i>LL_RTC_DisableIT_ALRA</i></a>            |
|          |        | <a href="#"><i>LL_RTC_EnableIT_ALRA</i></a>             |
|          |        | <a href="#"><i>LL_RTC_IsEnabledIT_ALRA</i></a>          |
|          | ALRBE  | <a href="#"><i>LL_RTC_ALMB_Disable</i></a>              |

| Register | Field | Function                                  |
|----------|-------|-------------------------------------------|
| ALRBIE   |       | <i>LL_RTC_ALMB_Enable</i>                 |
|          |       | <i>LL_RTC_DisableIT_ALRB</i>              |
|          |       | <i>LL_RTC_EnableIT_ALRB</i>               |
|          |       | <i>LL_RTC_IsEnabledIT_ALRB</i>            |
| BKP      |       | <i>LL_RTC_TIME_DisableDayLightStore</i>   |
|          |       | <i>LL_RTC_TIME_EnableDayLightStore</i>    |
|          |       | <i>LL_RTC_TIME_IsDayLightStoreEnabled</i> |
| BYPSHAD  |       | <i>LL_RTC_DisableShadowRegBypass</i>      |
|          |       | <i>LL_RTC_EnableShadowRegBypass</i>       |
|          |       | <i>LL_RTC_IsShadowRegBypassEnabled</i>    |
| COE      |       | <i>LL_RTC_CAL_GetOutputFreq</i>           |
|          |       | <i>LL_RTC_CAL_SetOutputFreq</i>           |
| COSEL    |       | <i>LL_RTC_CAL_GetOutputFreq</i>           |
|          |       | <i>LL_RTC_CAL_SetOutputFreq</i>           |
| DCE      |       | <i>LL_RTC_CAL_DisableCoarseDigital</i>    |
|          |       | <i>LL_RTC_CAL_EnableCoarseDigital</i>     |
| FMT      |       | <i>LL_RTC_GetHourFormat</i>               |
|          |       | <i>LL_RTC_SetHourFormat</i>               |
| OSEL     |       | <i>LL_RTC_GetAlarmOutEvent</i>            |
|          |       | <i>LL_RTC_SetAlarmOutEvent</i>            |
| POL      |       | <i>LL_RTC_GetOutputPolarity</i>           |
|          |       | <i>LL_RTC_SetOutputPolarity</i>           |
| REFCKON  |       | <i>LL_RTC_DisableRefClock</i>             |
|          |       | <i>LL_RTC_EnableRefClock</i>              |
| SUB1H    |       | <i>LL_RTC_TIME_DecHour</i>                |
| TSE      |       | <i>LL_RTC_TS_Disable</i>                  |
|          |       | <i>LL_RTC_TS_Enable</i>                   |
| TSEDGE   |       | <i>LL_RTC_TS_GetActiveEdge</i>            |
|          |       | <i>LL_RTC_TS_SetActiveEdge</i>            |
| TSIE     |       | <i>LL_RTC_DisableIT_TS</i>                |
|          |       | <i>LL_RTC_EnableIT_TS</i>                 |
|          |       | <i>LL_RTC_IsEnabledIT_TS</i>              |
| WUCKSEL  |       | <i>LL_RTC_WAKEUP_GetClock</i>             |
|          |       | <i>LL_RTC_WAKEUP_SetClock</i>             |
| WUTE     |       | <i>LL_RTC_WAKEUP_Disable</i>              |
|          |       | <i>LL_RTC_WAKEUP_Enable</i>               |

| Register | Field  | Function                         |
|----------|--------|----------------------------------|
| DR       | WUTIE  | <i>LL_RTC_WAKEUP_IsEnabled</i>   |
|          |        | <i>LL_RTC_DisableIT_WUT</i>      |
|          |        | <i>LL_RTC_EnableIT_WUT</i>       |
|          |        | <i>LL_RTC_IsEnabledIT_WUT</i>    |
|          | DT     | <i>LL_RTC_DATE_Config</i>        |
|          |        | <i>LL_RTC_DATE_Get</i>           |
|          |        | <i>LL_RTC_DATE_GetDay</i>        |
|          |        | <i>LL_RTC_DATE_SetDay</i>        |
|          | DU     | <i>LL_RTC_DATE_Config</i>        |
|          |        | <i>LL_RTC_DATE_Get</i>           |
|          |        | <i>LL_RTC_DATE_GetDay</i>        |
|          |        | <i>LL_RTC_DATE_SetDay</i>        |
|          | MT     | <i>LL_RTC_DATE_Config</i>        |
|          |        | <i>LL_RTC_DATE_Get</i>           |
|          |        | <i>LL_RTC_DATE_GetMonth</i>      |
|          |        | <i>LL_RTC_DATE_SetMonth</i>      |
|          | MU     | <i>LL_RTC_DATE_Config</i>        |
|          |        | <i>LL_RTC_DATE_Get</i>           |
|          |        | <i>LL_RTC_DATE_GetMonth</i>      |
|          |        | <i>LL_RTC_DATE_SetMonth</i>      |
|          | WDU    | <i>LL_RTC_DATE_Config</i>        |
|          |        | <i>LL_RTC_DATE_Get</i>           |
|          |        | <i>LL_RTC_DATE_GetWeekDay</i>    |
|          |        | <i>LL_RTC_DATE_SetWeekDay</i>    |
|          | YT     | <i>LL_RTC_DATE_Config</i>        |
|          |        | <i>LL_RTC_DATE_Get</i>           |
|          |        | <i>LL_RTC_DATE_GetYear</i>       |
|          |        | <i>LL_RTC_DATE_SetYear</i>       |
|          | YU     | <i>LL_RTC_DATE_Config</i>        |
|          |        | <i>LL_RTC_DATE_Get</i>           |
|          |        | <i>LL_RTC_DATE_GetYear</i>       |
|          |        | <i>LL_RTC_DATE_SetYear</i>       |
| ISR      | ALRAF  | <i>LL_RTC_ClearFlag_ALRA</i>     |
|          |        | <i>LL_RTC_IsActiveFlag_ALRA</i>  |
|          | ALRAWF | <i>LL_RTC_IsActiveFlag_ALRAW</i> |
|          | ALRBF  | <i>LL_RTC_ClearFlag_ALRB</i>     |

| Register | Field        | Function                                   |
|----------|--------------|--------------------------------------------|
|          | ALRBWF       | <a href="#">LL_RTC_IsActiveFlag_ALRB</a>   |
|          | INIT         | <a href="#">LL_RTC_DisableInitMode</a>     |
|          | INITF        | <a href="#">LL_RTC_EnableInitMode</a>      |
|          | INITS        | <a href="#">LL_RTC_IsActiveFlag_INITS</a>  |
|          | RECALPF      | <a href="#">LL_RTC_IsActiveFlag_RECALP</a> |
|          | RSF          | <a href="#">LL_RTC_ClearFlag_RS</a>        |
|          | SHPF         | <a href="#">LL_RTC_IsActiveFlag_SHP</a>    |
|          | TAMP1F       | <a href="#">LL_RTC_ClearFlag_TAMP1</a>     |
|          | TAMP2F       | <a href="#">LL_RTC_IsActiveFlag_TAMP1</a>  |
|          | TAMP3F       | <a href="#">LL_RTC_ClearFlag_TAMP2</a>     |
|          | TSF          | <a href="#">LL_RTC_IsActiveFlag_TAMP2</a>  |
|          | TSOVF        | <a href="#">LL_RTC_ClearFlag_TS</a>        |
|          | WUTF         | <a href="#">LL_RTC_IsActiveFlag_TS</a>     |
|          | WUTWF        | <a href="#">LL_RTC_ClearFlag_TSOV</a>      |
|          | PREDIV_A     | <a href="#">LL_RTC_IsActiveFlag_TSOV</a>   |
|          | PREDIV_S     | <a href="#">LL_RTC_ClearFlag_WUT</a>       |
|          | PRER         | <a href="#">LL_RTC_SetAsynchPrescaler</a>  |
|          | ADD1S        | <a href="#">LL_RTC_SetSynchPrescaler</a>   |
|          | SUBFS        | <a href="#">LL_RTC_GetAsynchPrescaler</a>  |
| SSR      | SS           | <a href="#">LL_RTC_SetSynchPrescaler</a>   |
| TAFCR    | ALARMOUTTYPE | <a href="#">LL_RTC_GetSubSecond</a>        |
|          | TAMP1E       | <a href="#">LL_RTC_SetAlarmOutputType</a>  |
|          | TAMP1TRG     | <a href="#">LL_RTC_Disable</a>             |
|          |              | <a href="#">LL_RTC_Enable</a>              |
|          |              | <a href="#">LL_RTC_DisableActiveLevel</a>  |
|          |              | <a href="#">LL_RTC_EnableActiveLevel</a>   |

| Register | Field     | Function                                |
|----------|-----------|-----------------------------------------|
| TR       | TAMP2E    | <i>LL_RTC_TAMPER_Disable</i>            |
|          |           | <i>LL_RTC_TAMPER_Enable</i>             |
|          | TAMP2TRG  | <i>LL_RTC_TAMPER_DisableActiveLevel</i> |
|          |           | <i>LL_RTC_TAMPER_EnableActiveLevel</i>  |
|          | TAMP3E    | <i>LL_RTC_TAMPER_Disable</i>            |
|          |           | <i>LL_RTC_TAMPER_Enable</i>             |
|          | TAMP3TRG  | <i>LL_RTC_TAMPER_DisableActiveLevel</i> |
|          |           | <i>LL_RTC_TAMPER_EnableActiveLevel</i>  |
|          | TAMPFLT   | <i>LL_RTC_TAMPER_GetFilterCount</i>     |
|          |           | <i>LL_RTC_TAMPER_SetFilterCount</i>     |
|          | TAMPFREQ  | <i>LL_RTC_TAMPER_GetSamplingFreq</i>    |
|          |           | <i>LL_RTC_TAMPER_SetSamplingFreq</i>    |
|          | TAMPIE    | <i>LL_RTC_DisableIT_TAMP</i>            |
|          |           | <i>LL_RTC_EnableIT_TAMP</i>             |
|          |           | <i>LL_RTC_IsEnabledIT_TAMP</i>          |
|          | TAMPPRCH  | <i>LL_RTC_TAMPER_GetPrecharge</i>       |
|          |           | <i>LL_RTC_TAMPER_SetPrecharge</i>       |
|          | TAMPPUDIS | <i>LL_RTC_TAMPER_DisablePullUp</i>      |
|          |           | <i>LL_RTC_TAMPER_EnablePullUp</i>       |
|          | TAMPTS    | <i>LL_RTC_TS_DisableOnTamper</i>        |
|          |           | <i>LL_RTC_TS_EnableOnTamper</i>         |
| TR       | HT        | <i>LL_RTC_TIME_Config</i>               |
|          |           | <i>LL_RTC_TIME_Get</i>                  |
|          |           | <i>LL_RTC_TIME_GetHour</i>              |
|          |           | <i>LL_RTC_TIME_SetHour</i>              |
|          | HU        | <i>LL_RTC_TIME_Config</i>               |
|          |           | <i>LL_RTC_TIME_Get</i>                  |
|          |           | <i>LL_RTC_TIME_GetHour</i>              |
|          |           | <i>LL_RTC_TIME_SetHour</i>              |
|          | MNT       | <i>LL_RTC_TIME_Config</i>               |
|          |           | <i>LL_RTC_TIME_Get</i>                  |
|          |           | <i>LL_RTC_TIME_GetMinute</i>            |
|          |           | <i>LL_RTC_TIME_SetMinute</i>            |
|          | MNU       | <i>LL_RTC_TIME_Config</i>               |
|          |           | <i>LL_RTC_TIME_Get</i>                  |
|          |           | <i>LL_RTC_TIME_GetMinute</i>            |

| Register | Field | Function                       |
|----------|-------|--------------------------------|
| TSDR     | PM    | <i>LL_RTC_TIME_SetMinute</i>   |
|          |       | <i>LL_RTC_TIME_Config</i>      |
|          |       | <i>LL_RTC_TIME_GetFormat</i>   |
|          |       | <i>LL_RTC_TIME_SetFormat</i>   |
|          | ST    | <i>LL_RTC_TIME_Config</i>      |
|          |       | <i>LL_RTC_TIME_Get</i>         |
|          |       | <i>LL_RTC_TIME_GetSecond</i>   |
|          |       | <i>LL_RTC_TIME_SetSecond</i>   |
|          | SU    | <i>LL_RTC_TIME_Config</i>      |
|          |       | <i>LL_RTC_TIME_Get</i>         |
|          |       | <i>LL_RTC_TIME_GetSecond</i>   |
|          |       | <i>LL_RTC_TIME_SetSecond</i>   |
| TSTR     | DT    | <i>LL_RTC_TS_GetDate</i>       |
|          |       | <i>LL_RTC_TS_GetDay</i>        |
|          | DU    | <i>LL_RTC_TS_GetDate</i>       |
|          |       | <i>LL_RTC_TS_GetDay</i>        |
|          | MT    | <i>LL_RTC_TS_GetDate</i>       |
|          |       | <i>LL_RTC_TS_GetMonth</i>      |
|          | MU    | <i>LL_RTC_TS_GetDate</i>       |
|          |       | <i>LL_RTC_TS_GetMonth</i>      |
|          | WDU   | <i>LL_RTC_TS_GetDate</i>       |
|          |       | <i>LL_RTC_TS_GetWeekDay</i>    |
| TSSSR    | SS    | <i>LL_RTC_TS_GetSubSecond</i>  |
| TSTR     | HT    | <i>LL_RTC_TS_GetHour</i>       |
|          |       | <i>LL_RTC_TS_GetTime</i>       |
|          | HU    | <i>LL_RTC_TS_GetHour</i>       |
|          |       | <i>LL_RTC_TS_GetTime</i>       |
|          | MNT   | <i>LL_RTC_TS_GetMinute</i>     |
|          |       | <i>LL_RTC_TS_GetTime</i>       |
|          | MNU   | <i>LL_RTC_TS_GetMinute</i>     |
|          |       | <i>LL_RTC_TS_GetTime</i>       |
|          | PM    | <i>LL_RTC_TS_GetTimeFormat</i> |
|          | ST    | <i>LL_RTC_TS_GetSecond</i>     |
|          |       | <i>LL_RTC_TS_GetTime</i>       |
|          | SU    | <i>LL_RTC_TS_GetSecond</i>     |
|          |       | <i>LL_RTC_TS_GetTime</i>       |

| Register | Field | Function                                             |
|----------|-------|------------------------------------------------------|
| WPR      | KEY   | <a href="#"><i>LL_RTC_DisableWriteProtection</i></a> |
|          |       | <a href="#"><i>LL_RTC_EnableWriteProtection</i></a>  |
| WUTR     | WUT   | <a href="#"><i>LL_RTC_WAKEUP_GetAutoReload</i></a>   |
|          |       | <a href="#"><i>LL_RTC_WAKEUP_SetAutoReload</i></a>   |

## 70.17 SPI

**Table 41: Correspondence between SPI registers and SPI low-layer driver functions**

| Register | Field    | Function                                           |
|----------|----------|----------------------------------------------------|
| CR1      | BIDIMODE | <a href="#"><i>LL_SPI_GetTransferDirection</i></a> |
|          |          | <a href="#"><i>LL_SPI_SetTransferDirection</i></a> |
|          | BIDIOE   | <a href="#"><i>LL_SPI_GetTransferDirection</i></a> |
|          |          | <a href="#"><i>LL_SPI_SetTransferDirection</i></a> |
|          | BR       | <a href="#"><i>LL_SPI_GetBaudRatePrescaler</i></a> |
|          |          | <a href="#"><i>LL_SPI_SetBaudRatePrescaler</i></a> |
|          | CPHA     | <a href="#"><i>LL_SPI_GetClockPhase</i></a>        |
|          |          | <a href="#"><i>LL_SPI_SetClockPhase</i></a>        |
|          | CPOL     | <a href="#"><i>LL_SPI_GetClockPolarity</i></a>     |
|          |          | <a href="#"><i>LL_SPI_SetClockPolarity</i></a>     |
|          | CRCEN    | <a href="#"><i>LL_SPI_DisableCRC</i></a>           |
|          |          | <a href="#"><i>LL_SPI_EnableCRC</i></a>            |
|          |          | <a href="#"><i>LL_SPI_IsEnabledCRC</i></a>         |
|          | CRCNEXT  | <a href="#"><i>LL_SPI_SetCRCNext</i></a>           |
|          | DFF      | <a href="#"><i>LL_SPI_GetDataWidth</i></a>         |
|          |          | <a href="#"><i>LL_SPI_SetDataWidth</i></a>         |
|          | LSBFIRST | <a href="#"><i>LL_SPI_GetTransferBitOrder</i></a>  |
|          |          | <a href="#"><i>LL_SPI_SetTransferBitOrder</i></a>  |
|          | MSTR     | <a href="#"><i>LL_SPI_GetMode</i></a>              |
|          |          | <a href="#"><i>LL_SPI_SetMode</i></a>              |
|          | RXONLY   | <a href="#"><i>LL_SPI_GetTransferDirection</i></a> |
|          |          | <a href="#"><i>LL_SPI_SetTransferDirection</i></a> |
|          | SPE      | <a href="#"><i>LL_SPI_Disable</i></a>              |
|          |          | <a href="#"><i>LL_SPI_Enable</i></a>               |
|          |          | <a href="#"><i>LL_SPI_IsEnabled</i></a>            |
|          | SSI      | <a href="#"><i>LL_SPI_GetMode</i></a>              |
|          |          | <a href="#"><i>LL_SPI_SetMode</i></a>              |
|          | SSM      | <a href="#"><i>LL_SPI_GetNSSMode</i></a>           |

| Register | Field   | Function                                          |
|----------|---------|---------------------------------------------------|
| CR2      | ERRIE   | <a href="#"><i>LL_SPI_SetNSSMode</i></a>          |
|          |         | <a href="#"><i>LL_SPI_DisableIT_ERR</i></a>       |
|          |         | <a href="#"><i>LL_SPI_EnableIT_ERR</i></a>        |
|          |         | <a href="#"><i>LL_SPI_IsEnabledIT_ERR</i></a>     |
|          | FRF     | <a href="#"><i>LL_SPI_GetStandard</i></a>         |
|          |         | <a href="#"><i>LL_SPI_SetStandard</i></a>         |
|          | RXDMAEN | <a href="#"><i>LL_SPI_DisableDMAReq_RX</i></a>    |
|          |         | <a href="#"><i>LL_SPI_EnableDMAReq_RX</i></a>     |
|          |         | <a href="#"><i>LL_SPI_IsEnabledDMAReq_RX</i></a>  |
|          | RXNEIE  | <a href="#"><i>LL_SPI_DisableIT_RXNE</i></a>      |
|          |         | <a href="#"><i>LL_SPI_EnableIT_RXNE</i></a>       |
|          |         | <a href="#"><i>LL_SPI_IsEnabledIT_RXNE</i></a>    |
|          | SSOE    | <a href="#"><i>LL_SPI_GetNSSMode</i></a>          |
|          |         | <a href="#"><i>LL_SPI_SetNSSMode</i></a>          |
|          |         | <a href="#"><i>LL_SPI_DisableDMAReq_TX</i></a>    |
|          | TXDMAEN | <a href="#"><i>LL_SPI_EnableDMAReq_TX</i></a>     |
|          |         | <a href="#"><i>LL_SPI_IsEnabledDMAReq_TX</i></a>  |
|          |         | <a href="#"><i>LL_SPI_DisableIT_TXE</i></a>       |
|          | TXEIE   | <a href="#"><i>LL_SPI_EnableIT_TXE</i></a>        |
|          |         | <a href="#"><i>LL_SPI_IsEnabledIT_TXE</i></a>     |
|          |         | <a href="#"><i>LL_SPI_DisableIT_BSY</i></a>       |
| CRCPR    | CRCPOLY | <a href="#"><i>LL_SPI_GetCRCPolynomial</i></a>    |
| CRCPR    | CRCPOLY | <a href="#"><i>LL_SPI_SetCRCPolynomial</i></a>    |
| DR       | DR      | <a href="#"><i>LL_SPI_DMA_GetRegAddr</i></a>      |
|          |         | <a href="#"><i>LL_SPI_ReceiveData16</i></a>       |
|          |         | <a href="#"><i>LL_SPI_ReceiveData8</i></a>        |
|          |         | <a href="#"><i>LL_SPI_TransmitData16</i></a>      |
|          |         | <a href="#"><i>LL_SPI_TransmitData8</i></a>       |
| RXCRCR   | RXCRC   | <a href="#"><i>LL_SPI_GetRxCRC</i></a>            |
| SR       | BSY     | <a href="#"><i>LL_SPI_IsActiveFlag_BSY</i></a>    |
|          | CRCERR  | <a href="#"><i>LL_SPI_ClearFlag_CRCERR</i></a>    |
|          |         | <a href="#"><i>LL_SPI_IsActiveFlag_CRCERR</i></a> |
|          | FRE     | <a href="#"><i>LL_SPI_ClearFlag_FRE</i></a>       |
|          |         | <a href="#"><i>LL_SPI_IsActiveFlag_FRE</i></a>    |
|          | MODF    | <a href="#"><i>LL_SPI_ClearFlag_MODF</i></a>      |
|          |         | <a href="#"><i>LL_SPI_IsActiveFlag_MODF</i></a>   |
|          | OVR     | <a href="#"><i>LL_SPI_ClearFlag_OVR</i></a>       |

| Register | Field | Function                                        |
|----------|-------|-------------------------------------------------|
|          |       | <a href="#"><i>LL_SPI_IsActiveFlag_OVR</i></a>  |
|          | RXNE  | <a href="#"><i>LL_SPI_IsActiveFlag_RXNE</i></a> |
|          | TXE   | <a href="#"><i>LL_SPI_IsActiveFlag_TXE</i></a>  |
| TXCRCR   | TXCRC | <a href="#"><i>LL_SPI_GetTxCRC</i></a>          |

## 70.18 SYSTEM

Table 42: Correspondence between SYSTEM registers and SYSTEM low-layer driver functions

| Register | Field                      | Function                                                                                                             |
|----------|----------------------------|----------------------------------------------------------------------------------------------------------------------|
| APB1_FZ  | DBG_I2C1_SMBUS_TIMEO<br>UT | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_I2C2_SMBUS_TIMEO<br>UT | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_IWDG_STOP              | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_RTC_STOP               | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_TIM2_STOP              | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_TIM3_STOP              | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_TIM4_STOP              | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_TIM5_STOP              | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_TIM6_STOP              | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_TIM7_STOP              | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
|          | DBG_WWDG_STOP              | <a href="#"><i>LL_DBGMCU_APB1_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB1_GRP1_UnFreezePeriph</i></a> |
| APB2_FZ  | DBG_TIM10_STOP             | <a href="#"><i>LL_DBGMCU_APB2_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB2_GRP1_UnFreezePeriph</i></a> |
|          | DBG_TIM11_STOP             | <a href="#"><i>LL_DBGMCU_APB2_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB2_GRP1_UnFreezePeriph</i></a> |
|          | DBG_TIM9_STOP              | <a href="#"><i>LL_DBGMCU_APB2_GRP1_FreezePeriph</i></a><br><a href="#"><i>LL_DBGMCU_APB2_GRP1_UnFreezePeriph</i></a> |

| Register      | Field       | Function                                          |
|---------------|-------------|---------------------------------------------------|
| DBGMCU_CR     | DBG_SLEEP   | <a href="#">LL_DBGMCU_DisableDBGSleepMode</a>     |
|               |             | <a href="#">LL_DBGMCU_EnableDBGSleepMode</a>      |
|               | DBG_STANDBY | <a href="#">LL_DBGMCU_DisableDBGStandbyMode</a>   |
|               |             | <a href="#">LL_DBGMCU_EnableDBGStandbyMode</a>    |
|               | DBG_STOP    | <a href="#">LL_DBGMCU_DisableDBGStopMode</a>      |
|               |             | <a href="#">LL_DBGMCU_EnableDBGStopMode</a>       |
|               | TRACE_IOEN  | <a href="#">LL_DBGMCU_GetTracePinAssignment</a>   |
|               |             | <a href="#">LL_DBGMCU_SetTracePinAssignment</a>   |
|               | TRACE_MODE  | <a href="#">LL_DBGMCU_GetTracePinAssignment</a>   |
|               |             | <a href="#">LL_DBGMCU_SetTracePinAssignment</a>   |
| DBGMCU_IDCODE | DEV_ID      | <a href="#">LL_DBGMCU_GetDeviceID</a>             |
|               | REV_ID      | <a href="#">LL_DBGMCU_GetRevisionID</a>           |
| FLASH_ACR     | ACC64       | <a href="#">LL_FLASH_Disable64bitAccess</a>       |
|               |             | <a href="#">LL_FLASH_Enable64bitAccess</a>        |
|               |             | <a href="#">LL_FLASH_Is64bitAccessEnabled</a>     |
|               | LATENCY     | <a href="#">LL_FLASH_GetLatency</a>               |
|               |             | <a href="#">LL_FLASH_SetLatency</a>               |
|               | PRFTEN      | <a href="#">LL_FLASH_DisablePrefetch</a>          |
|               |             | <a href="#">LL_FLASH_EnablePrefetch</a>           |
|               |             | <a href="#">LL_FLASH_IsPrefetchEnabled</a>        |
|               | RUN_PD      | <a href="#">LL_FLASH_DisableRunPowerDown</a>      |
|               |             | <a href="#">LL_FLASH_EnableRunPowerDown</a>       |
| FLASH_PDKEYR  | PDKEY1      | <a href="#">LL_FLASH_DisableRunPowerDown</a>      |
|               |             | <a href="#">LL_FLASH_EnableRunPowerDown</a>       |
|               | PDKEY2      | <a href="#">LL_FLASH_DisableRunPowerDown</a>      |
|               |             | <a href="#">LL_FLASH_EnableRunPowerDown</a>       |
| RI_ASCR1      | CH          | <a href="#">LL_RI_CloseIOSwitchLinkedToADC</a>    |
|               |             | <a href="#">LL_RI_OpenIOSwitchLinkedToADC</a>     |
|               | SCM         | <a href="#">LL_RI_DisableSwitchControlMode</a>    |
|               |             | <a href="#">LL_RI_EnableSwitchControlMode</a>     |
|               | VCOMP       | <a href="#">LL_RI_CloseIOSwitchLinkedToADC</a>    |
|               |             | <a href="#">LL_RI_OpenIOSwitchLinkedToADC</a>     |
| RI_ASCR2      | CH0b        | <a href="#">LL_RI_CloseIOSwitchNotLinkedToADC</a> |
|               |             | <a href="#">LL_RI_OpenIOSwitchNotLinkedToADC</a>  |

| Register | Field  | Function                                 |
|----------|--------|------------------------------------------|
|          | CH10b  | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH11b  | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH12b  | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH1b   | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH2b   | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH3b   | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH6b   | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH7b   | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH8b   | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | CH9b   | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR10_1 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR10_2 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR10_3 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR10_4 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR4_1  | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR4_2  | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR4_3  | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR4_4  | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |        | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |

| Register | Field | Function                                 |
|----------|-------|------------------------------------------|
|          | GR5_1 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |       | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR5_2 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |       | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR5_3 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |       | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR6_1 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |       | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR6_2 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |       | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR6_3 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |       | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
|          | GR6_4 | <i>LL_RI_CloseIOSwitchNotLinkedToADC</i> |
|          |       | <i>LL_RI_OpenIOSwitchNotLinkedToADC</i>  |
| RI_ASMR1 | PA    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PB    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PC    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PF    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PG    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
| RI_ASMR2 | PA    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PB    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PC    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PF    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PG    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
| RI_ASMR3 | PA    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |

| Register | Field | Function                                 |
|----------|-------|------------------------------------------|
| RI_ASMR4 | PB    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PC    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PF    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PG    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PA    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PB    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
| RI_ASMR5 | PC    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PF    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PG    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PA    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
|          | PB    | <i>LL_RI_ControlSwitchByADC</i>          |
|          |       | <i>LL_RI_ControlSwitchByTIM</i>          |
| RI_CICR1 | PC    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PB    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PA    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PF    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |

| Register | Field | Function                                 |
|----------|-------|------------------------------------------|
| RI_CICR2 | PG    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PA    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PB    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PC    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PF    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
| RI_CICR3 | PG    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PA    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PB    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PC    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PF    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
| RI_CICR4 | PG    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PA    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PB    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PC    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PF    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
| RI_CICR5 | PG    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PA    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |
|          | PB    | <i>LL_RI_IdentifyChannelIO</i>           |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i> |

| Register | Field | Function                                    |
|----------|-------|---------------------------------------------|
| RI_CMR1  | PC    | <i>LL_RI_IdentifyChannelIO</i>              |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i>    |
|          | PF    | <i>LL_RI_IdentifyChannelIO</i>              |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i>    |
|          | PG    | <i>LL_RI_IdentifyChannelIO</i>              |
|          |       | <i>LL_RI_IdentifySamplingCapacitorIO</i>    |
|          | PA    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PB    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PC    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
| RI_CMR2  | PF    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PG    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PA    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PB    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PC    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PF    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
| RI_CMR3  | PG    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PA    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PB    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PC    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PF    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|          | PG    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|          |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |

| Register  | Field | Function                                    |
|-----------|-------|---------------------------------------------|
| RI_CMRR4  | PA    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|           | PB    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|           | PC    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|           | PF    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|           | PG    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
| RI_CMRR5  | PA    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|           | PB    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|           | PC    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|           | PF    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
|           | PG    | <i>LL_RI_MaskChannelDuringAcquisition</i>   |
|           |       | <i>LL_RI_UnmaskChannelDuringAcquisition</i> |
| RI_HYSCR1 | PA    | <i>LL_RI_DisableHysteresis</i>              |
|           |       | <i>LL_RI_EnableHysteresis</i>               |
|           | PB    | <i>LL_RI_DisableHysteresis</i>              |
|           |       | <i>LL_RI_EnableHysteresis</i>               |
|           | PC    | <i>LL_RI_DisableHysteresis</i>              |
|           |       | <i>LL_RI_EnableHysteresis</i>               |
|           | PD    | <i>LL_RI_DisableHysteresis</i>              |
|           |       | <i>LL_RI_EnableHysteresis</i>               |
|           | PE    | <i>LL_RI_DisableHysteresis</i>              |
|           |       | <i>LL_RI_EnableHysteresis</i>               |
|           | PF    | <i>LL_RI_DisableHysteresis</i>              |
|           |       | <i>LL_RI_EnableHysteresis</i>               |
|           | PG    | <i>LL_RI_DisableHysteresis</i>              |
|           |       | <i>LL_RI_EnableHysteresis</i>               |
| RI_HYSCR2 | PA    | <i>LL_RI_DisableHysteresis</i>              |
|           |       | <i>LL_RI_EnableHysteresis</i>               |

| Register  | Field | Function                       |
|-----------|-------|--------------------------------|
| RI_HYSCR3 | PB    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PC    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PD    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PE    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PF    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PG    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
| RI_HYSCR4 | PA    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PB    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PC    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PD    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PE    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PF    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PG    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
| RI_HYSCR4 | PA    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PB    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PC    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PD    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |
|           | PE    | <i>LL_RI_DisableHysteresis</i> |
|           |       | <i>LL_RI_EnableHysteresis</i>  |

| Register        | Field  | Function                                  |
|-----------------|--------|-------------------------------------------|
| RI_ICR          | PF     | <i>LL_RI_DisableHysteresis</i>            |
|                 |        | <i>LL_RI_EnableHysteresis</i>             |
|                 | PG     | <i>LL_RI_DisableHysteresis</i>            |
|                 |        | <i>LL_RI_EnableHysteresis</i>             |
|                 | IC1    | <i>LL_RI_DisableRemapInputCapture_TIM</i> |
|                 |        | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
|                 | IC1OS  | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
|                 | IC2    | <i>LL_RI_DisableRemapInputCapture_TIM</i> |
|                 |        | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
|                 | IC2OS  | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
|                 | IC3    | <i>LL_RI_DisableRemapInputCapture_TIM</i> |
|                 |        | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
|                 | IC3OS  | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
|                 | IC4    | <i>LL_RI_DisableRemapInputCapture_TIM</i> |
|                 |        | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
|                 | IC4OS  | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
|                 | TIM    | <i>LL_RI_SetRemapInputCapture_TIM</i>     |
| SYSCFG_EXTICR_1 | EXTI0  | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI1  | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI10 | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI11 | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI12 | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI13 | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI14 | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI15 | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI2  | <i>LL_SYSCFG_GetEXTISource</i>            |
|                 |        | <i>LL_SYSCFG_SetEXTISource</i>            |
|                 | EXTI3  | <i>LL_SYSCFG_GetEXTISource</i>            |

| Register                   | Field  | Function                       |
|----------------------------|--------|--------------------------------|
| SYSCFG_EXTICR <sub>2</sub> | EXTI4  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI5  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI6  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI7  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI8  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI9  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI0  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI1  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI10 | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI11 | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI12 | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI13 | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI14 | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI15 | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI2  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI3  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI4  | <i>LL_SYSCFG_SetEXTISource</i> |
|                            |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                            | EXTI5  | <i>LL_SYSCFG_SetEXTISource</i> |

| Register           | Field  | Function                       |
|--------------------|--------|--------------------------------|
| SYSCFG_EXTICR<br>3 | EXTI6  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI7  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI8  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI9  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI10 | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI11 | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI12 | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI13 | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI14 | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI15 | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI2  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI3  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI4  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI5  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI6  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI7  | <i>LL_SYSCFG_SetEXTISource</i> |

| Register           | Field  | Function                       |
|--------------------|--------|--------------------------------|
| SYSCFG_EXTICR<br>4 | EXTI8  | <i>LL_SYSCFG_SetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI9  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI0  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI1  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI10 | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI11 | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI12 | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI13 | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI14 | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI15 | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI2  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI3  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI4  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI5  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI6  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI7  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI8  | <i>LL_SYSCFG_GetEXTISource</i> |
|                    |        | <i>LL_SYSCFG_SetEXTISource</i> |
|                    | EXTI9  | <i>LL_SYSCFG_GetEXTISource</i> |

| Register     | Field     | Function                                                  |
|--------------|-----------|-----------------------------------------------------------|
| SYSCFG_MEMRP |           | <a href="#">LL_SYSCFG_SetEXTISource</a>                   |
|              | BOOT_MODE | <a href="#">LL_SYSCFG_GetBootMode</a>                     |
|              | MEM_MODE  | <a href="#">LL_SYSCFG_GetRemapMemory</a>                  |
|              |           | <a href="#">LL_SYSCFG_SetRemapMemory</a>                  |
| SYSCFG_PMC   | LCD_CAPA  | <a href="#">LL_SYSCFG_DisableLCDCapacitanceConnection</a> |
|              |           | <a href="#">LL_SYSCFG_EnableLCDCapacitanceConnection</a>  |
|              | USB_PU    | <a href="#">LL_SYSCFG_DisableUSBPullUp</a>                |
|              |           | <a href="#">LL_SYSCFG_EnableUSBPullUp</a>                 |

## 70.19 TIM

**Table 43: Correspondence between TIM registers and TIM low-layer driver functions**

| Register | Field | Function                                   |
|----------|-------|--------------------------------------------|
| ARR      | ARR   | <a href="#">LL_TIM_GetAutoReload</a>       |
|          |       | <a href="#">LL_TIM_SetAutoReload</a>       |
| CCER     | CC1E  | <a href="#">LL_TIM_CC_DisableChannel</a>   |
|          |       | <a href="#">LL_TIM_CC_EnableChannel</a>    |
|          |       | <a href="#">LL_TIM_CC_IsEnabledChannel</a> |
|          | CC1NP | <a href="#">LL_TIM_IC_Config</a>           |
|          |       | <a href="#">LL_TIM_IC_GetPolarity</a>      |
|          |       | <a href="#">LL_TIM_IC_SetPolarity</a>      |
|          | CC1P  | <a href="#">LL_TIM_IC_Config</a>           |
|          |       | <a href="#">LL_TIM_IC_GetPolarity</a>      |
|          |       | <a href="#">LL_TIM_IC_SetPolarity</a>      |
|          |       | <a href="#">LL_TIM_OC_ConfigOutput</a>     |
|          |       | <a href="#">LL_TIM_OC_GetPolarity</a>      |
|          |       | <a href="#">LL_TIM_OC_SetPolarity</a>      |
|          | CC2E  | <a href="#">LL_TIM_CC_DisableChannel</a>   |
|          |       | <a href="#">LL_TIM_CC_EnableChannel</a>    |
|          |       | <a href="#">LL_TIM_CC_IsEnabledChannel</a> |
|          | CC2NP | <a href="#">LL_TIM_IC_Config</a>           |
|          |       | <a href="#">LL_TIM_IC_GetPolarity</a>      |
|          |       | <a href="#">LL_TIM_IC_SetPolarity</a>      |
|          | CC2P  | <a href="#">LL_TIM_IC_Config</a>           |
|          |       | <a href="#">LL_TIM_IC_GetPolarity</a>      |

| Register | Field | Function                          |
|----------|-------|-----------------------------------|
|          |       | <i>LL_TIM_IC_SetPolarity</i>      |
|          |       | <i>LL_TIM_OC_ConfigOutput</i>     |
|          |       | <i>LL_TIM_OC_GetPolarity</i>      |
|          |       | <i>LL_TIM_OC_SetPolarity</i>      |
|          | CC3E  | <i>LL_TIM_CC_DisableChannel</i>   |
|          |       | <i>LL_TIM_CC_EnableChannel</i>    |
|          |       | <i>LL_TIM_CC_IsEnabledChannel</i> |
|          | CC3NP | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetPolarity</i>      |
|          |       | <i>LL_TIM_IC_SetPolarity</i>      |
|          | CC3P  | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetPolarity</i>      |
|          |       | <i>LL_TIM_IC_SetPolarity</i>      |
|          |       | <i>LL_TIM_OC_ConfigOutput</i>     |
|          |       | <i>LL_TIM_OC_GetPolarity</i>      |
|          |       | <i>LL_TIM_OC_SetPolarity</i>      |
|          | CC4E  | <i>LL_TIM_CC_DisableChannel</i>   |
|          |       | <i>LL_TIM_CC_EnableChannel</i>    |
|          |       | <i>LL_TIM_CC_IsEnabledChannel</i> |
|          | CC4NP | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetPolarity</i>      |
|          |       | <i>LL_TIM_IC_SetPolarity</i>      |
|          | CC4P  | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetPolarity</i>      |
|          |       | <i>LL_TIM_IC_SetPolarity</i>      |
|          |       | <i>LL_TIM_OC_ConfigOutput</i>     |
|          |       | <i>LL_TIM_OC_GetPolarity</i>      |
|          |       | <i>LL_TIM_OC_SetPolarity</i>      |
| CCMR1    | CC1S  | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetActiveInput</i>   |
|          |       | <i>LL_TIM_IC_SetActiveInput</i>   |
|          |       | <i>LL_TIM_OC_ConfigOutput</i>     |
|          | CC2S  | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetActiveInput</i>   |
|          |       | <i>LL_TIM_IC_SetActiveInput</i>   |
|          |       | <i>LL_TIM_OC_ConfigOutput</i>     |

| Register | Field  | Function                          |
|----------|--------|-----------------------------------|
|          | IC1F   | <i>LL_TIM_IC_Config</i>           |
|          |        | <i>LL_TIM_IC_GetFilter</i>        |
|          |        | <i>LL_TIM_IC_SetFilter</i>        |
|          | IC1PSC | <i>LL_TIM_IC_Config</i>           |
|          |        | <i>LL_TIM_IC_GetPrescaler</i>     |
|          |        | <i>LL_TIM_IC_SetPrescaler</i>     |
|          | IC2F   | <i>LL_TIM_IC_Config</i>           |
|          |        | <i>LL_TIM_IC_GetFilter</i>        |
|          |        | <i>LL_TIM_IC_SetFilter</i>        |
|          | IC2PSC | <i>LL_TIM_IC_Config</i>           |
|          |        | <i>LL_TIM_IC_GetPrescaler</i>     |
|          |        | <i>LL_TIM_IC_SetPrescaler</i>     |
|          | OC1CE  | <i>LL_TIM_OC_DisableClear</i>     |
|          |        | <i>LL_TIM_OC_EnableClear</i>      |
|          |        | <i>LL_TIM_OC_IsEnabledClear</i>   |
|          | OC1FE  | <i>LL_TIM_OC_DisableFast</i>      |
|          |        | <i>LL_TIM_OC_EnableFast</i>       |
|          |        | <i>LL_TIM_OC_IsEnabledFast</i>    |
|          | OC1M   | <i>LL_TIM_OC_GetMode</i>          |
|          |        | <i>LL_TIM_OC_SetMode</i>          |
|          | OC1PE  | <i>LL_TIM_OC_DisablePreload</i>   |
|          |        | <i>LL_TIM_OC_EnablePreload</i>    |
|          |        | <i>LL_TIM_OC_IsEnabledPreload</i> |
|          | OC2CE  | <i>LL_TIM_OC_DisableClear</i>     |
|          |        | <i>LL_TIM_OC_EnableClear</i>      |
|          |        | <i>LL_TIM_OC_IsEnabledClear</i>   |
|          | OC2FE  | <i>LL_TIM_OC_DisableFast</i>      |
|          |        | <i>LL_TIM_OC_EnableFast</i>       |
|          |        | <i>LL_TIM_OC_IsEnabledFast</i>    |
|          | OC2M   | <i>LL_TIM_OC_GetMode</i>          |
|          |        | <i>LL_TIM_OC_SetMode</i>          |
|          | OC2PE  | <i>LL_TIM_OC_DisablePreload</i>   |
|          |        | <i>LL_TIM_OC_EnablePreload</i>    |
|          |        | <i>LL_TIM_OC_IsEnabledPreload</i> |
| CCMR2    | CC3S   | <i>LL_TIM_IC_Config</i>           |
|          |        | <i>LL_TIM_IC_GetActiveInput</i>   |

| Register | Field | Function                          |
|----------|-------|-----------------------------------|
|          |       | <i>LL_TIM_IC_SetActiveInput</i>   |
|          |       | <i>LL_TIM_OC_ConfigOutput</i>     |
| CC4S     |       | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetActiveInput</i>   |
|          |       | <i>LL_TIM_IC_SetActiveInput</i>   |
|          |       | <i>LL_TIM_OC_ConfigOutput</i>     |
| IC3F     |       | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetFilter</i>        |
|          |       | <i>LL_TIM_IC_SetFilter</i>        |
| IC3PSC   |       | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetPrescaler</i>     |
|          |       | <i>LL_TIM_IC_SetPrescaler</i>     |
| IC4F     |       | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetFilter</i>        |
|          |       | <i>LL_TIM_IC_SetFilter</i>        |
| IC4PSC   |       | <i>LL_TIM_IC_Config</i>           |
|          |       | <i>LL_TIM_IC_GetPrescaler</i>     |
|          |       | <i>LL_TIM_IC_SetPrescaler</i>     |
| OC3CE    |       | <i>LL_TIM_OC_DisableClear</i>     |
|          |       | <i>LL_TIM_OC_EnableClear</i>      |
|          |       | <i>LL_TIM_OC_IsEnabledClear</i>   |
| OC3FE    |       | <i>LL_TIM_OC_DisableFast</i>      |
|          |       | <i>LL_TIM_OC_EnableFast</i>       |
|          |       | <i>LL_TIM_OC_IsEnabledFast</i>    |
| OC3M     |       | <i>LL_TIM_OC_GetMode</i>          |
|          |       | <i>LL_TIM_OC_SetMode</i>          |
| OC3PE    |       | <i>LL_TIM_OC_DisablePreload</i>   |
|          |       | <i>LL_TIM_OC_EnablePreload</i>    |
|          |       | <i>LL_TIM_OC_IsEnabledPreload</i> |
| OC4CE    |       | <i>LL_TIM_OC_DisableClear</i>     |
|          |       | <i>LL_TIM_OC_EnableClear</i>      |
|          |       | <i>LL_TIM_OC_IsEnabledClear</i>   |
| OC4FE    |       | <i>LL_TIM_OC_DisableFast</i>      |
|          |       | <i>LL_TIM_OC_EnableFast</i>       |
|          |       | <i>LL_TIM_OC_IsEnabledFast</i>    |
| OC4M     |       | <i>LL_TIM_OC_GetMode</i>          |

| Register | Field | Function                           |
|----------|-------|------------------------------------|
|          | OC4PE | <i>LL_TIM_OC_SetMode</i>           |
|          |       | <i>LL_TIM_OC_DisablePreload</i>    |
|          |       | <i>LL_TIM_OC_EnablePreload</i>     |
|          |       | <i>LL_TIM_OC_IsEnabledPreload</i>  |
| CCR1     | CCR1  | <i>LL_TIM_IC_GetCaptureCH1</i>     |
|          |       | <i>LL_TIM_OC_GetCompareCH1</i>     |
|          |       | <i>LL_TIM_OC_SetCompareCH1</i>     |
| CCR2     | CCR2  | <i>LL_TIM_IC_GetCaptureCH2</i>     |
|          |       | <i>LL_TIM_OC_GetCompareCH2</i>     |
|          |       | <i>LL_TIM_OC_SetCompareCH2</i>     |
| CCR3     | CCR3  | <i>LL_TIM_IC_GetCaptureCH3</i>     |
|          |       | <i>LL_TIM_OC_GetCompareCH3</i>     |
|          |       | <i>LL_TIM_OC_SetCompareCH3</i>     |
| CCR4     | CCR4  | <i>LL_TIM_IC_GetCaptureCH4</i>     |
|          |       | <i>LL_TIM_OC_GetCompareCH4</i>     |
|          |       | <i>LL_TIM_OC_SetCompareCH4</i>     |
| CNT      | CNT   | <i>LL_TIM_GetCounter</i>           |
|          |       | <i>LL_TIM_SetCounter</i>           |
| CR1      | ARPE  | <i>LL_TIM_DisableARRPreload</i>    |
|          |       | <i>LL_TIM_EnableARRPreload</i>     |
|          |       | <i>LL_TIM_IsEnabledARRPreload</i>  |
|          | CEN   | <i>LL_TIM_DisableCounter</i>       |
|          |       | <i>LL_TIM_EnableCounter</i>        |
|          |       | <i>LL_TIM_IsEnabledCounter</i>     |
|          | CKD   | <i>LL_TIM_GetClockDivision</i>     |
|          |       | <i>LL_TIM_SetClockDivision</i>     |
|          | CMS   | <i>LL_TIM_GetCounterMode</i>       |
|          |       | <i>LL_TIM_SetCounterMode</i>       |
|          | DIR   | <i>LL_TIM_GetCounterMode</i>       |
|          |       | <i>LL_TIM_GetDirection</i>         |
|          |       | <i>LL_TIM_SetCounterMode</i>       |
|          | OPM   | <i>LL_TIM_GetOnePulseMode</i>      |
|          |       | <i>LL_TIM_SetOnePulseMode</i>      |
|          | UDIS  | <i>LL_TIM_DisableUpdateEvent</i>   |
|          |       | <i>LL_TIM_EnableUpdateEvent</i>    |
|          |       | <i>LL_TIM_IsEnabledUpdateEvent</i> |

| Register | Field | Function                                 |
|----------|-------|------------------------------------------|
|          | URS   | <i>LL_TIM_GetUpdateSource</i>            |
|          |       | <i>LL_TIM_SetUpdateSource</i>            |
| CR2      | CCDS  | <i>LL_TIM_CC_GetDMAReqTrigger</i>        |
|          |       | <i>LL_TIM_CC_SetDMAReqTrigger</i>        |
|          | MMS   | <i>LL_TIM_SetTriggerOutput</i>           |
|          | TI1S  | <i>LL_TIM_IC_DisableXORCombination</i>   |
|          |       | <i>LL_TIM_IC_EnableXORCombination</i>    |
|          |       | <i>LL_TIM_IC_IsEnabledXORCombination</i> |
| DCR      | DBA   | <i>LL_TIM_ConfigDMAburst</i>             |
|          | DBL   | <i>LL_TIM_ConfigDMAburst</i>             |
| DIER     | CC1DE | <i>LL_TIM_DisableDMAReq_CC1</i>          |
|          |       | <i>LL_TIM_EnableDMAReq_CC1</i>           |
|          |       | <i>LL_TIM_IsEnabledDMAReq_CC1</i>        |
|          | CC1IE | <i>LL_TIM_DisableIT_CC1</i>              |
|          |       | <i>LL_TIM_EnableIT_CC1</i>               |
|          |       | <i>LL_TIM_IsEnabledIT_CC1</i>            |
|          | CC2DE | <i>LL_TIM_DisableDMAReq_CC2</i>          |
|          |       | <i>LL_TIM_EnableDMAReq_CC2</i>           |
|          |       | <i>LL_TIM_IsEnabledDMAReq_CC2</i>        |
|          | CC2IE | <i>LL_TIM_DisableIT_CC2</i>              |
|          |       | <i>LL_TIM_EnableIT_CC2</i>               |
|          |       | <i>LL_TIM_IsEnabledIT_CC2</i>            |
|          | CC3DE | <i>LL_TIM_DisableDMAReq_CC3</i>          |
|          |       | <i>LL_TIM_EnableDMAReq_CC3</i>           |
|          |       | <i>LL_TIM_IsEnabledDMAReq_CC3</i>        |
|          | CC3IE | <i>LL_TIM_DisableIT_CC3</i>              |
|          |       | <i>LL_TIM_EnableIT_CC3</i>               |
|          |       | <i>LL_TIM_IsEnabledIT_CC3</i>            |
|          | CC4DE | <i>LL_TIM_DisableDMAReq_CC4</i>          |
|          |       | <i>LL_TIM_EnableDMAReq_CC4</i>           |
|          |       | <i>LL_TIM_IsEnabledDMAReq_CC4</i>        |
|          | CC4IE | <i>LL_TIM_DisableIT_CC4</i>              |
|          |       | <i>LL_TIM_EnableIT_CC4</i>               |
|          |       | <i>LL_TIM_IsEnabledIT_CC4</i>            |
|          | TDE   | <i>LL_TIM_DisableDMAReq_TRIG</i>         |
|          |       | <i>LL_TIM_EnableDMAReq_TRIG</i>          |

| Register | Field | Function                               |
|----------|-------|----------------------------------------|
| TIE      | TIE   | <i>LL_TIM_IsEnabledDMAReq_TRIG</i>     |
|          |       | <i>LL_TIM_DisableIT_TRIG</i>           |
|          |       | <i>LL_TIM_EnableIT_TRIG</i>            |
|          |       | <i>LL_TIM_IsEnabledIT_TRIG</i>         |
|          | UDE   | <i>LL_TIM_DisableDMAReq_UPDATE</i>     |
|          |       | <i>LL_TIM_EnableDMAReq_UPDATE</i>      |
|          |       | <i>LL_TIM_IsEnabledDMAReq_UPDATE</i>   |
|          | UIE   | <i>LL_TIM_DisableIT_UPDATE</i>         |
|          |       | <i>LL_TIM_EnableIT_UPDATE</i>          |
|          |       | <i>LL_TIM_IsEnabledIT_UPDATE</i>       |
| EGR      | CC1G  | <i>LL_TIM_GenerateEvent_CC1</i>        |
|          | CC2G  | <i>LL_TIM_GenerateEvent_CC2</i>        |
|          | CC3G  | <i>LL_TIM_GenerateEvent_CC3</i>        |
|          | CC4G  | <i>LL_TIM_GenerateEvent_CC4</i>        |
|          | TG    | <i>LL_TIM_GenerateEvent_TRIG</i>       |
|          | UG    | <i>LL_TIM_GenerateEvent_UPDATE</i>     |
| PSC      | PSC   | <i>LL_TIM_GetPrescaler</i>             |
|          |       | <i>LL_TIM_SetPrescaler</i>             |
| SMCR     | ECE   | <i>LL_TIM_DisableExternalClock</i>     |
|          |       | <i>LL_TIM_EnableExternalClock</i>      |
|          |       | <i>LL_TIM_IsEnabledExternalClock</i>   |
|          |       | <i>LL_TIM_SetClockSource</i>           |
|          | ETF   | <i>LL_TIM_ConfigETR</i>                |
|          | ETP   | <i>LL_TIM_ConfigETR</i>                |
|          | ETPS  | <i>LL_TIM_ConfigETR</i>                |
|          | MSM   | <i>LL_TIM_DisableMasterSlaveMode</i>   |
|          |       | <i>LL_TIM_EnableMasterSlaveMode</i>    |
|          |       | <i>LL_TIM_IsEnabledMasterSlaveMode</i> |
|          | OCCS  | <i>LL_TIM_SetOCRefClearInputSource</i> |
|          | SMS   | <i>LL_TIM_SetClockSource</i>           |
|          |       | <i>LL_TIM_SetEncoderMode</i>           |
|          |       | <i>LL_TIM_SetSlaveMode</i>             |
|          | TS    | <i>LL_TIM_SetTriggerInput</i>          |
| SR       | CC1IF | <i>LL_TIM_ClearFlag_CC1</i>            |
|          |       | <i>LL_TIM_IsActiveFlag_CC1</i>         |
|          | CC1OF | <i>LL_TIM_ClearFlag_CC1OVR</i>         |

| Register | Field      | Function                          |
|----------|------------|-----------------------------------|
|          | CC2IF      | <i>LL_TIM_IsActiveFlag_CC1OVR</i> |
|          |            | <i>LL_TIM_ClearFlag_CC2</i>       |
|          |            | <i>LL_TIM_IsActiveFlag_CC2</i>    |
|          | CC2OF      | <i>LL_TIM_ClearFlag_CC2OVR</i>    |
|          |            | <i>LL_TIM_IsActiveFlag_CC2OVR</i> |
|          | CC3IF      | <i>LL_TIM_ClearFlag_CC3</i>       |
|          |            | <i>LL_TIM_IsActiveFlag_CC3</i>    |
|          | CC3OF      | <i>LL_TIM_ClearFlag_CC3OVR</i>    |
|          |            | <i>LL_TIM_IsActiveFlag_CC3OVR</i> |
|          | CC4IF      | <i>LL_TIM_ClearFlag_CC4</i>       |
|          |            | <i>LL_TIM_IsActiveFlag_CC4</i>    |
|          | CC4OF      | <i>LL_TIM_ClearFlag_CC4OVR</i>    |
|          |            | <i>LL_TIM_IsActiveFlag_CC4OVR</i> |
|          | TIF        | <i>LL_TIM_ClearFlag_TRIG</i>      |
|          |            | <i>LL_TIM_IsActiveFlag_TRIG</i>   |
|          | UIF        | <i>LL_TIM_ClearFlag_UPDATE</i>    |
|          |            | <i>LL_TIM_IsActiveFlag_UPDATE</i> |
| TIM10_OR | ETR_RMP    | <i>LL_TIM_SetRemap</i>            |
|          | TI1_RMP    | <i>LL_TIM_SetRemap</i>            |
|          | TI1_RMP_RI | <i>LL_TIM_SetRemap</i>            |
| TIM11_OR | ETR_RMP    | <i>LL_TIM_SetRemap</i>            |
|          | TI1_RMP    | <i>LL_TIM_SetRemap</i>            |
|          | TI1_RMP_RI | <i>LL_TIM_SetRemap</i>            |
| TIM2_OR  | ITR1_RMP   | <i>LL_TIM_SetRemap</i>            |
| TIM3_OR  | ITR2_RMP   | <i>LL_TIM_SetRemap</i>            |
| TIM9_OR  | ITR1_RMP   | <i>LL_TIM_SetRemap</i>            |
|          | TI1_RMP    | <i>LL_TIM_SetRemap</i>            |

## 70.20 USART

Table 44: Correspondence between USART registers and USART low-layer driver functions

| Register | Field  | Function                         |
|----------|--------|----------------------------------|
| BRR      | BRR    | <i>LL_USART_GetBaudRate</i>      |
|          |        | <i>LL_USART_SetBaudRate</i>      |
| CR1      | IDLEIE | <i>LL_USART_DisableIT_IDLE</i>   |
|          |        | <i>LL_USART_EnableIT_IDLE</i>    |
|          |        | <i>LL_USART_IsEnabledIT_IDLE</i> |

| Register | Field  | Function                             |
|----------|--------|--------------------------------------|
|          | M      | <i>LL_USART_ConfigCharacter</i>      |
|          |        | <i>LL_USART_GetDataWidth</i>         |
|          |        | <i>LL_USART_SetDataWidth</i>         |
|          | OVER8  | <i>LL_USART_GetOverSampling</i>      |
|          |        | <i>LL_USART_SetOverSampling</i>      |
|          | PCE    | <i>LL_USART_ConfigCharacter</i>      |
|          |        | <i>LL_USART_GetParity</i>            |
|          |        | <i>LL_USART_SetParity</i>            |
|          | PEIE   | <i>LL_USART_DisableIT_PE</i>         |
|          |        | <i>LL_USART_EnableIT_PE</i>          |
|          |        | <i>LL_USART_IsEnabledIT_PE</i>       |
|          | PS     | <i>LL_USART_ConfigCharacter</i>      |
|          |        | <i>LL_USART_GetParity</i>            |
|          |        | <i>LL_USART_SetParity</i>            |
|          | RE     | <i>LL_USART_DisableDirectionRx</i>   |
|          |        | <i>LL_USART_EnableDirectionRx</i>    |
|          |        | <i>LL_USART_GetTransferDirection</i> |
|          |        | <i>LL_USART_SetTransferDirection</i> |
|          | RWU    | <i>LL_USART_IsActiveFlag_RWU</i>     |
|          |        | <i>LL_USART_RequestEnterMuteMode</i> |
|          |        | <i>LL_USART_RequestExitMuteMode</i>  |
|          | RXNEIE | <i>LL_USART_DisableIT_RXNE</i>       |
|          |        | <i>LL_USART_EnableIT_RXNE</i>        |
|          |        | <i>LL_USART_IsEnabledIT_RXNE</i>     |
|          | SBK    | <i>LL_USART_IsActiveFlag_SBK</i>     |
|          |        | <i>LL_USART_RequestBreakSending</i>  |
|          | TCIE   | <i>LL_USART_DisableIT_TC</i>         |
|          |        | <i>LL_USART_EnableIT_TC</i>          |
|          |        | <i>LL_USART_IsEnabledIT_TC</i>       |
|          | TE     | <i>LL_USART_DisableDirectionTx</i>   |
|          |        | <i>LL_USART_EnableDirectionTx</i>    |
|          |        | <i>LL_USART_GetTransferDirection</i> |
|          |        | <i>LL_USART_SetTransferDirection</i> |
|          | TXEIE  | <i>LL_USART_DisableIT_TXE</i>        |
|          |        | <i>LL_USART_EnableIT_TXE</i>         |
|          |        | <i>LL_USART_IsEnabledIT_TXE</i>      |

| Register | Field | Function                                               |
|----------|-------|--------------------------------------------------------|
| CR2      | UE    | <a href="#"><i>LL_USART_Disable</i></a>                |
|          |       | <a href="#"><i>LL_USART_Enable</i></a>                 |
|          |       | <a href="#"><i>LL_USART_IsEnabled</i></a>              |
|          | WAKE  | <a href="#"><i>LL_USART_GetWakeUpMethod</i></a>        |
|          |       | <a href="#"><i>LL_USART_SetWakeUpMethod</i></a>        |
|          | ADD   | <a href="#"><i>LL_USART_GetNodeAddress</i></a>         |
|          |       | <a href="#"><i>LL_USART_SetNodeAddress</i></a>         |
|          | CLKEN | <a href="#"><i>LL_USART_ConfigAsyncMode</i></a>        |
|          |       | <a href="#"><i>LL_USART_ConfigHalfDuplexMode</i></a>   |
|          |       | <a href="#"><i>LL_USART_ConfigIrdaMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_ConfigLINMode</i></a>          |
|          |       | <a href="#"><i>LL_USART_ConfigMultiProcessMode</i></a> |
|          |       | <a href="#"><i>LL_USART_ConfigSmartcardMode</i></a>    |
|          |       | <a href="#"><i>LL_USART_ConfigSyncMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_DisableSCLKOutput</i></a>      |
|          |       | <a href="#"><i>LL_USART_EnableSCLKOutput</i></a>       |
|          |       | <a href="#"><i>LL_USART_IsEnabledSCLKOutput</i></a>    |
|          | CPHA  | <a href="#"><i>LL_USART_ConfigClock</i></a>            |
|          |       | <a href="#"><i>LL_USART_GetClockPhase</i></a>          |
|          |       | <a href="#"><i>LL_USART_SetClockPhase</i></a>          |
|          | CPOL  | <a href="#"><i>LL_USART_ConfigClock</i></a>            |
|          |       | <a href="#"><i>LL_USART_GetClockPolarity</i></a>       |
|          |       | <a href="#"><i>LL_USART_SetClockPolarity</i></a>       |
|          | LBCL  | <a href="#"><i>LL_USART_ConfigClock</i></a>            |
|          |       | <a href="#"><i>LL_USART_GetLastClkPulseOutput</i></a>  |
|          |       | <a href="#"><i>LL_USART_SetLastClkPulseOutput</i></a>  |
|          | LBDIE | <a href="#"><i>LL_USART_DisableIT_LBD</i></a>          |
|          |       | <a href="#"><i>LL_USART_EnableIT_LBD</i></a>           |
|          |       | <a href="#"><i>LL_USART_IsEnabledIT_LBD</i></a>        |
|          | LBDL  | <a href="#"><i>LL_USART_GetLINBrkDetectionLen</i></a>  |
|          |       | <a href="#"><i>LL_USART_SetLINBrkDetectionLen</i></a>  |
|          | LINEN | <a href="#"><i>LL_USART_ConfigAsyncMode</i></a>        |
|          |       | <a href="#"><i>LL_USART_ConfigHalfDuplexMode</i></a>   |
|          |       | <a href="#"><i>LL_USART_ConfigIrdaMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_ConfigLINMode</i></a>          |
|          |       | <a href="#"><i>LL_USART_ConfigMultiProcessMode</i></a> |

| Register | Field | Function                                               |
|----------|-------|--------------------------------------------------------|
| CR3      | STOP  | <a href="#"><i>LL_USART_ConfigSmartcardMode</i></a>    |
|          |       | <a href="#"><i>LL_USART_ConfigSyncMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_DisableLIN</i></a>             |
|          |       | <a href="#"><i>LL_USART_EnableLIN</i></a>              |
|          |       | <a href="#"><i>LL_USART_IsEnabledLIN</i></a>           |
|          |       | <a href="#"><i>LL_USART_ConfigCharacter</i></a>        |
|          |       | <a href="#"><i>LL_USART_ConfigIrdaMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_ConfigLINMode</i></a>          |
|          |       | <a href="#"><i>LL_USART_ConfigSmartcardMode</i></a>    |
|          |       | <a href="#"><i>LL_USART_GetStopBitsLength</i></a>      |
|          |       | <a href="#"><i>LL_USART_SetStopBitsLength</i></a>      |
|          | CTSE  | <a href="#"><i>LL_USART_DisableCTSHWFlowCtrl</i></a>   |
|          |       | <a href="#"><i>LL_USART_EnableCTSHWFlowCtrl</i></a>    |
|          |       | <a href="#"><i>LL_USART_GetHWFlowCtrl</i></a>          |
|          |       | <a href="#"><i>LL_USART_SetHWFlowCtrl</i></a>          |
|          | CTSIE | <a href="#"><i>LL_USART_DisableIT_CTS</i></a>          |
|          |       | <a href="#"><i>LL_USART_EnableIT_CTS</i></a>           |
|          |       | <a href="#"><i>LL_USART_IsEnabledIT_CTS</i></a>        |
|          | DMAR  | <a href="#"><i>LL_USART_DisableDMAReq_RX</i></a>       |
|          |       | <a href="#"><i>LL_USART_EnableDMAReq_RX</i></a>        |
|          |       | <a href="#"><i>LL_USART_IsEnabledDMAReq_RX</i></a>     |
|          | DMAT  | <a href="#"><i>LL_USART_DisableDMAReq_TX</i></a>       |
|          |       | <a href="#"><i>LL_USART_EnableDMAReq_TX</i></a>        |
|          |       | <a href="#"><i>LL_USART_IsEnabledDMAReq_TX</i></a>     |
|          | EIE   | <a href="#"><i>LL_USART_DisableIT_ERROR</i></a>        |
|          |       | <a href="#"><i>LL_USART_EnableIT_ERROR</i></a>         |
|          |       | <a href="#"><i>LL_USART_IsEnabledIT_ERROR</i></a>      |
|          | HDSEL | <a href="#"><i>LL_USART_ConfigAsyncMode</i></a>        |
|          |       | <a href="#"><i>LL_USART_ConfigHalfDuplexMode</i></a>   |
|          |       | <a href="#"><i>LL_USART_ConfigIrdaMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_ConfigLINMode</i></a>          |
|          |       | <a href="#"><i>LL_USART_ConfigMultiProcessMode</i></a> |
|          |       | <a href="#"><i>LL_USART_ConfigSmartcardMode</i></a>    |
|          |       | <a href="#"><i>LL_USART_ConfigSyncMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_DisableHalfDuplex</i></a>      |
|          |       | <a href="#"><i>LL_USART_EnableHalfDuplex</i></a>       |

| Register | Field | Function                                               |
|----------|-------|--------------------------------------------------------|
| IREN     |       | <a href="#"><i>LL_USART_IsEnabledHalfDuplex</i></a>    |
|          |       | <a href="#"><i>LL_USART_ConfigAsyncMode</i></a>        |
|          |       | <a href="#"><i>LL_USART_ConfigHalfDuplexMode</i></a>   |
|          |       | <a href="#"><i>LL_USART_ConfigIrdaMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_ConfigLINMode</i></a>          |
|          |       | <a href="#"><i>LL_USART_ConfigMultiProcessMode</i></a> |
|          |       | <a href="#"><i>LL_USART_ConfigSyncMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_DisableIrda</i></a>            |
|          |       | <a href="#"><i>LL_USART_EnableIrda</i></a>             |
|          |       | <a href="#"><i>LL_USART_IsEnabledIrda</i></a>          |
| IRLP     |       | <a href="#"><i>LL_USART_GetIrdaPowerMode</i></a>       |
|          |       | <a href="#"><i>LL_USART_SetIrdaPowerMode</i></a>       |
| NACK     |       | <a href="#"><i>LL_USART_DisableSmartcardNACK</i></a>   |
|          |       | <a href="#"><i>LL_USART_EnableSmartcardNACK</i></a>    |
|          |       | <a href="#"><i>LL_USART_IsEnabledSmartcardNACK</i></a> |
| ONEBIT   |       | <a href="#"><i>LL_USART_DisableOneBitSamp</i></a>      |
|          |       | <a href="#"><i>LL_USART_EnableOneBitSamp</i></a>       |
|          |       | <a href="#"><i>LL_USART_IsEnabledOneBitSamp</i></a>    |
| RTSE     |       | <a href="#"><i>LL_USART_DisableRTSHWFlowCtrl</i></a>   |
|          |       | <a href="#"><i>LL_USART_EnableRTSHWFlowCtrl</i></a>    |
|          |       | <a href="#"><i>LL_USART_GetHWFlowCtrl</i></a>          |
|          |       | <a href="#"><i>LL_USART_SetHWFlowCtrl</i></a>          |
| SCEN     |       | <a href="#"><i>LL_USART_ConfigAsyncMode</i></a>        |
|          |       | <a href="#"><i>LL_USART_ConfigHalfDuplexMode</i></a>   |
|          |       | <a href="#"><i>LL_USART_ConfigIrdaMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_ConfigLINMode</i></a>          |
|          |       | <a href="#"><i>LL_USART_ConfigMultiProcessMode</i></a> |
|          |       | <a href="#"><i>LL_USART_ConfigSmartcardMode</i></a>    |
|          |       | <a href="#"><i>LL_USART_ConfigSyncMode</i></a>         |
|          |       | <a href="#"><i>LL_USART_DisableSmartcard</i></a>       |
|          |       | <a href="#"><i>LL_USART_EnableSmartcard</i></a>        |
|          |       | <a href="#"><i>LL_USART_IsEnabledSmartcard</i></a>     |
| DR       | DR    | <a href="#"><i>LL_USART_DMA_GetRegAddr</i></a>         |
|          |       | <a href="#"><i>LL_USART_ReceiveData8</i></a>           |
|          |       | <a href="#"><i>LL_USART_ReceiveData9</i></a>           |
|          |       | <a href="#"><i>LL_USART_TransmitData8</i></a>          |

| Register | Field | Function                                       |
|----------|-------|------------------------------------------------|
| GTPR     | GT    | <a href="#">LL_USART_TransmitData9</a>         |
|          |       | <a href="#">LL_USART_GetSmartcardGuardTime</a> |
|          |       | <a href="#">LL_USART_SetSmartcardGuardTime</a> |
|          | PSC   | <a href="#">LL_USART_GetIrdaPrescaler</a>      |
|          |       | <a href="#">LL_USART_GetSmartcardPrescaler</a> |
|          |       | <a href="#">LL_USART_SetIrdaPrescaler</a>      |
|          |       | <a href="#">LL_USART_SetSmartcardPrescaler</a> |
| SR       | CTS   | <a href="#">LL_USART_ClearFlag_nCTS</a>        |
|          |       | <a href="#">LL_USART_IsActiveFlag_nCTS</a>     |
|          | FE    | <a href="#">LL_USART_ClearFlag_FE</a>          |
|          |       | <a href="#">LL_USART_IsActiveFlag_FE</a>       |
|          | IDLE  | <a href="#">LL_USART_ClearFlag_IDLE</a>        |
|          |       | <a href="#">LL_USART_IsActiveFlag_IDLE</a>     |
|          | LBD   | <a href="#">LL_USART_ClearFlag_LBD</a>         |
|          |       | <a href="#">LL_USART_IsActiveFlag_LBD</a>      |
|          | NF    | <a href="#">LL_USART_ClearFlag_NE</a>          |
|          |       | <a href="#">LL_USART_IsActiveFlag_NE</a>       |
|          | ORE   | <a href="#">LL_USART_ClearFlag_ORE</a>         |
|          |       | <a href="#">LL_USART_IsActiveFlag_ORE</a>      |
|          | PE    | <a href="#">LL_USART_ClearFlag_PE</a>          |
|          |       | <a href="#">LL_USART_IsActiveFlag_PE</a>       |
|          | RXNE  | <a href="#">LL_USART_ClearFlag_RXNE</a>        |
|          |       | <a href="#">LL_USART_IsActiveFlag_RXNE</a>     |
|          | TC    | <a href="#">LL_USART_ClearFlag_TC</a>          |
|          |       | <a href="#">LL_USART_IsActiveFlag_TC</a>       |
|          | TXE   | <a href="#">LL_USART_IsActiveFlag_TXE</a>      |

## 70.21 WWDG

Table 45: Correspondence between WWDG registers and WWDG low-layer driver functions

| Register | Field | Function                                  |
|----------|-------|-------------------------------------------|
| CFR      | EWI   | <a href="#">LL_WWDG_EnableIT_EWKUP</a>    |
|          |       | <a href="#">LL_WWDG_IsEnabledIT_EWKUP</a> |
|          | W     | <a href="#">LL_WWDG_GetWindow</a>         |
|          |       | <a href="#">LL_WWDG_SetWindow</a>         |
|          | WDGTB | <a href="#">LL_WWDG_GetPrescaler</a>      |
|          |       | <a href="#">LL_WWDG_SetPrescaler</a>      |

| Register | Field | Function                          |
|----------|-------|-----------------------------------|
| CR       | T     | <i>LL_WWDG_GetCounter</i>         |
|          |       | <i>LL_WWDG_SetCounter</i>         |
|          | WDGA  | <i>LL_WWDG_Enable</i>             |
|          |       | <i>LL_WWDG_IsEnabled</i>          |
| SR       | EWIF  | <i>LL_WWDG_ClearFlag_EWKUP</i>    |
|          |       | <i>LL_WWDG_IsActiveFlag_EWKUP</i> |

# 71

# FAQs

## General subjects

### Why should I use the HAL drivers?

There are many advantages in using the HAL drivers:

- Ease of use: you can use the HAL drivers to configure and control any peripheral embedded within your STM32 MCU without prior in-depth knowledge of the product.
- HAL drivers provide intuitive and ready-to-use APIs to configure the peripherals and support polling, interrupt and DMA programming model to accommodate all application requirements, thus allowing the end-user to build a complete application by calling a few APIs.
- Higher level of abstraction than a standard peripheral library allowing to transparently manage:
  - Data transfers and processing using blocking mode (polling) or non-blocking mode (interrupt or DMA)
  - Error management through peripheral error detection and timeout mechanism.
- Generic architecture speeding up initialization and porting, thus allowing customers to focus on innovation.
- Generic set of APIs with full compatibility across the STM32 series/lines, to ease the porting task between STM32 MCUs.
- The APIs provided within the HAL drivers are feature-oriented and do not require in-depth knowledge of peripheral operation.
- The APIs provided are modular. They include initialization, IO operation and control functions. The end-user has to call init function, then start the process by calling one IO operation functions (write, read, transmit, receive, ...). Most of the peripherals have the same architecture.
- The number of functions required to build a complete and useful application is very reduced. As an example, to build a UART communication process, the user only has to call HAL\_UART\_Init() then HAL\_UART\_Transmit() or HAL\_UART\_Receive().

### Which STM32L1 devices are supported by the HAL drivers?

The HAL drivers are developed to support all STM32L1 devices. To ensure compatibility between all devices and portability with others series and lines, the API is split into the generic and the extension APIs . For more details, please refer to [Section 2.4: "Devices supported by HAL drivers"](#).

### What is the cost of using HAL drivers in term of code size and performance?

Like generic architecture drivers, the HAL drivers may induce firmware overhead.

This is due to the high abstraction level and ready-to-use APIs which allow data transfers, errors management and offloads the user application from implementation details.

## Architecture

### How many files should I modify to configure the HAL drivers?

Only one file needs to be modified: stm32l1xx\_hal\_conf.h. You can modify this file by disabling unused modules, or adjusting some parameters (i.e. HSE value, System configuration...)

A template is provided in the HAL driver folders (stm32l1xx\_hal\_conf\_template.c).

### Which header files should I include in my application to use the HAL drivers?

Only stm32l1xx\_hal.h file has to be included.

### What is the difference between `stm32l1xx_hal_ppp.c/h` and `stm32l1xx_hal_ppp_ex.c/h?`

The HAL driver architecture supports common features across STM32 series/lines. To support specific features, the drivers are split into two groups.

- The generic APIs (`stm32l1xx_hal_ppp.c`): It includes the common set of APIs across all the STM32 product lines
- The extension APIs (`stm32l1xx_hal_ppp_ex.c`): It includes the specific APIs for specific device part number or family.

### Initialization and I/O operation functions

#### How do I configure the system clock?

Unlike the standard library, the system clock configuration is not performed in CMSIS drivers file (`system_stm32l1xx.c`) but in the main user application by calling the two main functions, `HAL_RCC_OscConfig()` and `HAL_RCC_ClockConfig()`. It can be modified in any user application section.

#### What is the purpose of the `PPP_HandleTypeDef *pHandle` structure located in each driver in addition to the Initialization structure

`PPP_HandleTypeDef *pHandle` is the main structure implemented in the HAL drivers. It handles the peripheral configuration and registers, and embeds all the structures and variables required to follow the peripheral device flow (pointer to buffer, Error code, State,...)

However, this structure is not required to service peripherals such as GPIO, SYSTICK, PWR, and RCC.

#### What is the purpose of `HAL_PPP_MspInit()` and `HAL_PPP_MspDeInit()` functions?

These function are called within `HAL_PPP_Init()` and `HAL_PPP_DeInit()`, respectively. They are used to perform the low level Initialization/de-initialization related to the additional hardware resources (RCC, GPIO, NVIC and DMA).

These functions are declared in `stm32l1xx_hal_msp.c`. A template is provided in the HAL driver folders (`stm32l1xx_hal_msp_template.c`).

#### When and how should I use callbacks functions (functions declared with the attribute `__weak`)?

Use callback functions for the I/O operations used in DMA or interrupt mode. The PPP process complete callbacks are called to inform the user about process completion in real-time event mode (interrupts).

The Errors callbacks are called when a processing error occurs in DMA or interrupt mode. These callbacks are customized by the user to add user proprietary code. They can be declared in the application. Note that the same process completion callbacks are used for DMA and interrupt mode.

### **Is it mandatory to use HAL\_Init() function at the beginning of the user application?**

It is mandatory to use HAL\_Init() function to enable the system configuration (Prefetch, Data instruction cache,...), configure the systTick and the NVIC priority grouping and the hardware low level initialization.

The sysTick configuration shall be adjusted by calling **HAL\_RCC\_ClockConfig()** function, to obtain 1 ms whatever the system clock.

### **Why do I need to configure the SysTick timer to use the HAL drivers?**

The SysTick timer is configured to be used to generate variable increments by calling **HAL\_IncTick()** function in Systick ISR and retrieve the value of this variable by calling **HAL\_GetTick()** function.

The call **HAL\_GetTick()** function is mandatory when using HAL drivers with Polling Process or when using **HAL\_Delay()**.

### **Why is the SysTick timer configured to have 1 ms?**

This is mandatory to ensure correct IO operation in particular for polling mode operation where the 1 ms is required as timebase.

### **Could HAL\_Delay() function block my application under certain conditions?**

Care must be taken when using **HAL\_Delay()** since this function provides accurate delay based on a variable incremented in SysTick ISR. This implies that if **HAL\_Delay()** is called from a peripheral ISR process, then the SysTick interrupt must have higher priority (numerically lower) than the peripheral interrupt, otherwise the caller ISR process will be blocked. Use **HAL\_NVIC\_SetPriority()** function to change the SysTick interrupt priority.

### **What programming model sequence should I follow to use HAL drivers ?**

Follow the sequence below to use the APIs provided in the HAL drivers:

1. Call **HAL\_Init()** function to initialize the system (data cache, NVIC priority,...).
2. Initialize the system clock by calling **HAL\_RCC\_OscConfig()** followed by **HAL\_RCC\_ClockConfig()**.
3. Add **HAL\_IncTick()** function under **SysTick\_Handler()** ISR function to enable polling process when using **HAL\_Delay()** function
4. Start initializing your peripheral by calling **HAL\_PPP\_Init()**.
5. Implement the hardware low level initialization (Peripheral clock, GPIO, DMA,..) by calling **HAL\_PPP\_MspInit()** in **stm32l1xx\_hal\_msp.c**
6. Start your process operation by calling IO operation functions.

### **What is the purpose of HAL\_PPP\_IRQHandler() function and when should I use it?**

**HAL\_PPP\_IRQHandler()** is used to handle interrupt process. It is called under **PPP\_IRQHandler()** function in **stm32l1xx\_it.c**. In this case, the end-user has to implement only the callbacks functions (prefixed by **\_weak**) to perform the appropriate action when an interrupt is detected. Advanced users can implement their own code in **PPP\_IRQHandler()** without calling **HAL\_PPP\_IRQHandler()**.

### **Can I use directly the macros defined in **stm32l1xx\_hal\_ppp.h** ?**

Yes, you can: a set of macros is provided with the APIs. They allow accessing directly some specific features using peripheral flags.

**Where must PPP\_HandleTypeDef structure peripheral handler be declared?**

PPP\_HandleTypeDef structure peripheral handler must be declared as a global variable, so that all the structure fields are set to 0 by default. In this way, the peripheral handler default state are set to HAL\_PPP\_STATE\_RESET, which is the default state for each peripheral after a system reset.

## 72 Revision history

Table 46: Document revision history

| Date        | Revision | Changes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18-Nov-2014 | 1        | Initial release.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 13-Feb-2015 | 2        | <p>Updated list of PPP_TypeDef structures in <a href="#">Section 2.5.1: "HAL API naming rules"</a>.</p> <p>Updated <a href="#">Table 5: "List of devices supported by HAL drivers"</a>.</p> <p>Updated , and <a href="#">Section 18.1.2: "FLASH_OBProgramInitTypeDef"</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 21-Apr-2015 | 3        | <p>Changed path for compilation under Unix environment. Updated drivers to be C++ compliant.</p> <p>Updated common macros in <a href="#">Section 2.9: "HAL common resources"</a>.</p> <p>Added interface to access MPU features (refer to <code>stm32l1xx_hal_cortex.h</code>).</p> <p><a href="#">Section 12: "HAL CRYP Generic Driver"</a>: added instance field in CRYP_HandleTypeDef.</p> <p><a href="#">Section 17: "HAL FLASH Generic Driver"</a>: changed field name of NOR_CFITypeDef (CFI1X changed to CFI1_X).</p> <p><a href="#">Section 30: "HAL PCD Generic Driver"</a>:</p> <ul style="list-style-type: none"> <li>• HAL_PCD_ActiveRemoteWakeup renamed<br/>HAL_PCD_ActivateRemoteWakeup</li> <li>• HAL_PCD_DeActiveRemoteWakeup renamed to<br/>HAL_PCD_DeActivateRemoteWakeup</li> <li>• HAL_PCD_ActiveRemoteWakeup renamed<br/>HAL_PCD_ActivateRemoteWakeup</li> <li>• HAL_PCD_DeActiveRemoteWakeup renamed to<br/>HAL_PCD_DeActivateRemoteWakeup.</li> </ul> <p><a href="#">Section 32: "HAL PWR Generic Driver"</a>:</p> <ul style="list-style-type: none"> <li>– HAL_PWR_PVDConfig renamed HAL_PWR_ConfigPVD.</li> <li>– Added new interfaces: <ul style="list-style-type: none"> <li>– void HAL_PWR_EnableSleepOnExit(void);</li> <li>– void HAL_PWR_DisableSleepOnExit(void);</li> <li>– void HAL_PWR_EnableSEVOnPend(void);</li> <li>– void HAL_PWR_DisableSEVOnPend(void);</li> <li>– void HAL_PWR_EnableSleepOnExit(void);</li> <li>– uint32_t HAL_PWREx_GetVoltageRange(void);</li> </ul> </li> </ul> <p><a href="#">Section 34: "HAL RCC Generic Driver"</a>:</p> <ul style="list-style-type: none"> <li>• HAL_RCC_CCSCallback renamed to HAL_RCC_CSSCallback.</li> <li>• Added HAL_RCCEx_GetPeriphCLKFreq interface.</li> </ul> <p><a href="#">Section 39: "HAL SMARTCARD Generic Driver"</a>: Removed HAL_SMARTCARD_Relinit interface.</p> <p><a href="#">Section 40: "HAL SPI Generic Driver"</a>: HAL_SPI_GetError now returns a uint32_t instead of HAL_SPI_ErrorTypeDef.</p> <p><a href="#">Section 43: "HAL TIM Generic Driver"</a>: added HAL_TIM_SlaveConfigSynchronization_IT interface.</p> <p><a href="#">Section 45: "HAL UART Generic Driver"</a>: Changed ErrorCode field of UART_HandleTypeDef from HAL_UART_ErrorTypeDef to uint32_t.</p> <p><a href="#">Section 46: "HAL USART Generic Driver"</a>: Changed ErrorCode field of USART_HandleTypeDef from HAL_USART_ErrorTypeDef to uint32_t.</p> |

| Date        | Revision | Changes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7-Jul-2016  | 4        | <p>Added low-layer driver APIs.</p> <p><i>Section 16: "HAL DMA Generic Driver"</i>: added new HAL_DMA_Abort_IT function to abort DMA transfers in Interrupt mode.</p> <p><i>Section 25: "HAL IWDG Generic Driver"</i>: driver simplified by removing HAL_IWDG_Start(), HAL_IWDG_MspInit() and HAL_IWDG_GetState() APIs. API functions are:</p> <ul style="list-style-type: none"> <li>• HAL_IWDG_Init() function, which performs IWDG counter configuration and start</li> <li>• HAL_IWDG_Refresh() function, which performs the IWDG counter reloading.</li> </ul> <p><i>Section 47: "HAL WWDG Generic Driver"</i>:</p> <ul style="list-style-type: none"> <li>• Driver simplified by removing HAL_WWDG_Start(), HAL_WWDG_Start_IT(), HAL_WWDG_MspInit() and HAL_WWDG_GetState() APIs. API functions are: HAL_WWDG_Init(), HAL_WWDG_MspInit(), HAL_WWDG_Refresh(), HAL_WWDG_IRQHandler() and HAL_WWDG_EarlyWakeupCallback()</li> <li>• Updated HAL_WWDG_Refresh() API to remove counter parameter</li> <li>• Added new EWIMode field in WWDG_InitTypeDef to select Early Wakeup Interrupt.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 31-May-2017 | 5        | <p>Enhanced HAL delay and time base implementation: Added new templates stm32l1xx_hal_timebase_tim_template.c which can be used to override the native HAL time base functions (defined as weak) and to use Timer as time base tick source. Updated the following functions:</p> <ul style="list-style-type: none"> <li>• HAL Cortex: <ul style="list-style-type: none"> <li>– updated: Updated HAL_MPUI_Disable function to clear whole CR register</li> </ul> </li> <li>• LL DMA: <ul style="list-style-type: none"> <li>– Improved CPAR and CMAR registers access in LL_DMA_ConfigAddresses function</li> </ul> </li> <li>• HAL FLASH: <ul style="list-style-type: none"> <li>– Updated HAL_FLASHEx_Erase_IT() function to check the FLASH is ready before starting Erase by IT</li> </ul> </li> <li>• HAL/LL GPIO: <ul style="list-style-type: none"> <li>– Renamed GPIO_AFRL_AFRLx and GPIO_AFRL_AFRHx bit to GPIO_AFRL_AFSELx</li> </ul> </li> <li>• HAL/LL I2C: <ul style="list-style-type: none"> <li>– Added LL_I2C_DisableReset() function to allow the disable of SWRST</li> </ul> </li> <li>• HAL PCD: <ul style="list-style-type: none"> <li>– Corrected double buffer implementation in PCD_SET_EP_DBUF1_CNT() macro</li> <li>– Added missing USB_CNTR_SOFM in the setting of wInterrupt_Mask global variable used in HAL_PCD_Init</li> <li>– Removed lock/unlock from receive and transmit endpoints</li> </ul> </li> <li>• HAL/LL PWR: <ul style="list-style-type: none"> <li>– Replaced HAL_PWREx_GetVoltageRange() function by direct register access to remove dependency with HAL PWR</li> <li>– Renamed the LL_PWR_IsActiveFlag_VOSF() API to LL_PWR_IsActiveFlag_VOS() in order to remove reference to PWR flag name and to refer to the Power feature</li> </ul> </li> <li>• HAL/LL RTC: <ul style="list-style-type: none"> <li>– Renamed RTC_CR_BCK bits in RTC_CR register to RTC_CR_BKP, to be aligned with others series</li> </ul> </li> <li>• HAL/LL SPI: <ul style="list-style-type: none"> <li>– Removed LL_SPI_SR_UDR define which is available only for I2S feature</li> </ul> </li> </ul> |

| Date        | Revision | Changes                                                                                                                                                                                                                                                                            |
|-------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31-May-2017 | 5        | <ul style="list-style-type: none"><li>- Updated LL_SPI_TransmitData16() et LL_SPI_TransmitData8 functions</li><li>• HAL/LL TIM:<ul style="list-style-type: none"><li>- Corrected error in LL_TIM_EnableUpdateEvent() and LL_TIM_DisableUpdateEvent() functions</li></ul></li></ul> |

**IMPORTANT NOTICE – PLEASE 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 acknowledgement.

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. 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.

© 2017 STMicroelectronics – All rights reserved