

Description of STM32L1xx HAL drivers

Introduction

STMCube™ is an STMicroelectronics original initiative to ease developers life by reducing development efforts, time and cost. STM32Cube covers STM32 portfolio.

STM32Cube Version 1.x includes:

- The STM32CubeMX, a graphical software configuration tool that allows generating C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per series (such as STM32CubeL1 for STM32L1 series)
 - The STM32Cube HAL, an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio
 - A consistent set of middleware components such as RTOS, USB, TCP/IP, Graphics
 - All embedded software utilities coming with a full set of examples.

The HAL drivers layer provides a generic multi instance simple set of APIs (application programming interfaces) to interact with the upper layer (application, libraries and stacks). It is composed of generic and extension APIs. It is directly built around a generic architecture and allows the built-upon layers, such as the middleware layer, to implement their functions without knowing in-depth how to use the MCU. This structure improves the library code reusability and guarantees an easy portability on other devices.

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, to manage data transfers based on polling, to handle interrupts or DMA, and to manage communication errors.

The HAL drivers 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 family or part number.

The HAL drivers are feature-oriented instead of IP-oriented. As an example, the timer APIs are split into several categories following the functions offered by the IP: basic timer, capture, pulse width modulation (PWM), etc..

The drivers source code 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.

The HAL drivers 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.

This user manual is structured as follows:

- Overview of the HAL drivers
- Detailed description of each peripheral driver: configuration structures, functions, and how to use the given API to build your application.



Contents

1 Acronyms and definitions.....	35
2 Overview of HAL drivers	37
2.1 HAL and user-application files.....	37
2.1.1 HAL driver files	37
2.1.2 User-application files	38
2.2 HAL data structures	40
2.2.1 Peripheral handle structures	40
2.2.2 Initialization and configuration structure	42
2.2.3 Specific process structures	42
2.3 API classification	42
2.4 Devices supported by HAL drivers	44
2.5 HAL drivers rules.....	46
2.5.1 HAL API naming rules	46
2.5.2 HAL general naming rules.....	47
2.5.3 HAL interrupt handler and callback functions.....	48
2.6 HAL generic APIs.....	49
2.7 HAL extension APIs	50
2.7.1 HAL extension model overview	50
2.7.2 HAL extension model cases.....	51
2.8 File inclusion model.....	52
2.9 HAL common resources.....	53
2.10 HAL configuration.....	54
2.11 HAL system peripheral handling	55
2.11.1 Clock.....	55
2.11.2 GPIOs.....	56
2.11.3 Cortex NVIC and SysTick timer.....	57
2.11.4 PWR	58
2.11.5 EXTI.....	58
2.11.6 DMA.....	59
2.12 How to use HAL drivers	61
2.12.1 HAL usage models	61
2.12.2 HAL initialization	61
2.12.3 HAL IO operation process	63
2.12.4 Timeout and error management.....	66
3 HAL System Driver	71

3.1	HAL Firmware driver API description	71
3.1.1	How to use this driver	71
3.1.2	Initialization and de-initialization functions	71
3.1.3	HAL Control functions.....	71
3.1.4	HAL_Init.....	72
3.1.5	HAL_DelInit	72
3.1.6	HAL_MspInit.....	73
3.1.7	HAL_MspDeInit	73
3.1.8	HAL_InitTick	73
3.1.9	HAL_IncTick	74
3.1.10	HAL_GetTick	74
3.1.11	HAL_Delay	74
3.1.12	HAL_SuspendTick.....	75
3.1.13	HAL_ResumeTick.....	75
3.1.14	HAL_GetHalVersion	76
3.1.15	HAL_GetREVID	76
3.1.16	HAL_GetDEVID	76
3.1.17	HAL_EnableDBGSleepMode	76
3.1.18	HAL_DisableDBGSleepMode	77
3.1.19	HAL_EnableDBGStopMode	77
3.1.20	HAL_DisableDBGStopMode	77
3.1.21	HAL_EnableDBGStandbyMode	78
3.1.22	HAL_DisableDBGStandbyMode	78
3.2	HAL Firmware driver defines.....	78
3.2.1	HAL.....	78
4	HAL ADC Generic Driver.....	79
4.1	ADC Firmware driver registers structures	79
4.1.1	ADC_InitTypeDef.....	79
4.1.2	ADC_ChannelConfTypeDef	81
4.1.3	ADC_AnalogWDGConfTypeDef.....	82
4.1.4	ADC_HandleTypeDef.....	83
4.2	ADC Firmware driver API description.....	83
4.2.1	ADC specific features	83
4.2.2	How to use this driver	84
4.2.3	Initialization and de-initialization functions	84
4.2.4	IO operation functions	85
4.2.5	Peripheral Control functions	85
4.2.6	Peripheral State and Errors functions	85

4.2.7	HAL_ADC_Init	86
4.2.8	HAL_ADC_DelInit.....	86
4.2.9	HAL_ADC_MspInit	87
4.2.10	HAL_ADC_MspDeInit.....	87
4.2.11	HAL_ADC_Start	87
4.2.12	HAL_ADC_Stop.....	88
4.2.13	HAL_ADC_PollForConversion	88
4.2.14	HAL_ADC_PollForEvent	88
4.2.15	HAL_ADC_Start_IT	89
4.2.16	HAL_ADC_Stop_IT	89
4.2.17	HAL_ADC_Start_DMA	89
4.2.18	HAL_ADC_Stop_DMA.....	90
4.2.19	HAL_ADC_GetValue	90
4.2.20	HAL_ADC_IRQHandler	91
4.2.21	HAL_ADC_ConvCpltCallback	91
4.2.22	HAL_ADC_ConvHalfCpltCallback	91
4.2.23	HAL_ADC_LevelOutOfWindowCallback	92
4.2.24	HAL_ADC_ErrorCallback	92
4.2.25	HAL_ADC_ConfigChannel	92
4.2.26	HAL_ADC_AnalogWDGConfig	93
4.2.27	HAL_ADC_GetState.....	93
4.2.28	HAL_ADC_GetError	94
4.3	ADC Firmware driver defines	94
4.3.1	ADC	94
5	HAL ADC Extension Driver	101
5.1	ADCEx Firmware driver registers structures	101
5.1.1	ADC_InjectionConfTypeDef	101
5.2	ADCEx Firmware driver API description	102
5.2.1	How to use this driver	102
5.2.2	IO operation functions	103
5.2.3	Peripheral Control functions	103
5.2.4	HAL_ADCEx_InjectedStart	103
5.2.5	HAL_ADCEx_InjectedStop.....	104
5.2.6	HAL_ADCEx_InjectedPollForConversion	104
5.2.7	HAL_ADCEx_InjectedStart_IT	104
5.2.8	HAL_ADCEx_InjectedStop_IT	105
5.2.9	HAL_ADCEx_InjectedGetValue	105
5.2.10	HAL_ADCEx_InjectedConvCpltCallback	106

5.2.11	HAL_ADCEx_InjectedConfigChannel	106
5.3	ADCEx Firmware driver defines	106
5.3.1	ADCEx	106
6	HAL COMP Generic Driver.....	111
6.1	COMP Firmware driver registers structures	111
6.1.1	COMP_InitTypeDef	111
6.1.2	COMP_HandleTypeDef	112
6.2	COMP Firmware driver API description	112
6.2.1	COMP Peripheral features	112
6.2.2	How to use this driver	113
6.2.3	Initialization and de-initialization functions	114
6.2.4	IO operation functions	114
6.2.5	Peripheral Control functions	114
6.2.6	Peripheral State functions	114
6.2.7	HAL_COMP_Init	115
6.2.8	HAL_COMP_DeInit	115
6.2.9	HAL_COMP_MspInit	115
6.2.10	HAL_COMP_MspDeInit	116
6.2.11	HAL_COMP_Start	116
6.2.12	HAL_COMP_Stop	116
6.2.13	HAL_COMP_Start_IT	117
6.2.14	HAL_COMP_Stop_IT	117
6.2.15	HAL_COMP_IRQHandler	117
6.2.16	HAL_COMP_Lock	118
6.2.17	HAL_COMP_GetOutputLevel	118
6.2.18	HAL_COMP_TriggerCallback	118
6.2.19	HAL_COMP_GetState	119
6.3	COMP Firmware driver defines	119
6.3.1	COMP	119
7	HAL COMP Extension Driver.....	123
7.1	COMPEx Firmware driver defines.....	123
7.1.1	COMPEx	123
8	HAL CORTEX Generic Driver.....	125
8.1	CORTEX Firmware driver API description	125
8.1.1	Initialization and de-initialization functions	125
8.1.2	Peripheral Control functions	125
8.1.3	HAL_NVIC_SetPriorityGrouping	125

8.1.4	HAL_NVIC_SetPriority	126
8.1.5	HAL_NVIC_EnableIRQ	126
8.1.6	HAL_NVIC_DisableIRQ.....	127
8.1.7	HAL_NVIC_SystemReset.....	127
8.1.8	HAL_SYSTICK_Config.....	127
8.1.9	HAL_NVIC_GetPriorityGrouping	128
8.1.10	HAL_NVIC_GetPriority	128
8.1.11	HAL_NVIC_SetPendingIRQ	129
8.1.12	HAL_NVIC_GetPendingIRQ	129
8.1.13	HAL_NVIC_ClearPendingIRQ.....	130
8.1.14	HAL_NVIC_GetActive	130
8.1.15	HAL_SYSTICK_CLKSourceConfig	130
8.1.16	HAL_SYSTICK_IRQHandler	131
8.1.17	HAL_SYSTICK_Callback	131
8.2	CORTEX Firmware driver defines.....	131
8.2.1	CORTEX.....	131
9	HAL CRC Generic Driver.....	133
9.1	CRC Firmware driver registers structures	133
9.1.1	CRC_HandleTypeDef.....	133
9.2	CRC Firmware driver API description	133
9.2.1	How to use this driver	133
9.2.2	Initialization and de-initialization functions	133
9.2.3	Peripheral Control functions	134
9.2.4	Peripheral State functions	134
9.2.5	HAL_CRC_Init	134
9.2.6	HAL_CRC_DeInit	134
9.2.7	HAL_CRC_MspInit	135
9.2.8	HAL_CRC_MspDeInit.....	135
9.2.9	HAL_CRC_Accumulate.....	135
9.2.10	HAL_CRC_Calculate.....	136
9.2.11	HAL_CRC_GetState.....	136
9.2.12	HAL_CRC_Accumulate.....	137
9.2.13	HAL_CRC_Calculate.....	137
9.3	CRC Firmware driver defines	137
9.3.1	CRC	137
10	HAL CRYP Generic Driver.....	139
10.1	CRYP Firmware driver registers structures	139
10.1.1	CRYP_HandleTypeDef	139

10.1.2	CRYP_HandleTypeDef.....	139
10.2	CRYP Firmware driver API description	140
10.2.1	Initialization and de-initialization functions	140
10.2.2	AES processing functions	140
10.2.3	DMA callback functions	141
10.2.4	CRYP IRQ handler management.....	141
10.2.5	Peripheral State functions	141
10.2.6	HAL_CRYP_Init.....	141
10.2.7	HAL_CRYP_DelInit	141
10.2.8	HAL_CRYP_MspInit.....	142
10.2.9	HAL_CRYP_MspDelInit	142
10.2.10	HAL_CRYP_AESECB_Encrypt.....	142
10.2.11	HAL_CRYP_AESCBC_Encrypt	143
10.2.12	HAL_CRYP_AESCTR_Encrypt.....	143
10.2.13	HAL_CRYP_AESECB_Decrypt	144
10.2.14	HAL_CRYP_AESCBC_Decrypt	144
10.2.15	HAL_CRYP_AESCTR_Decrypt	145
10.2.16	HAL_CRYP_AESECB_Encrypt_IT	145
10.2.17	HAL_CRYP_AESCBC_Encrypt_IT	146
10.2.18	HAL_CRYP_AESCTR_Encrypt_IT	146
10.2.19	HAL_CRYP_AESECB_Decrypt_IT	147
10.2.20	HAL_CRYP_AESCBC_Decrypt_IT	147
10.2.21	HAL_CRYP_AESCTR_Decrypt_IT	148
10.2.22	HAL_CRYP_AESECB_Encrypt_DMA.....	148
10.2.23	HAL_CRYP_AESCBC_Encrypt_DMA	149
10.2.24	HAL_CRYP_AESCTR_Encrypt_DMA.....	149
10.2.25	HAL_CRYP_AESECB_Decrypt_DMA	150
10.2.26	HAL_CRYP_AESCBC_Decrypt_DMA	150
10.2.27	HAL_CRYP_AESCTR_Decrypt_DMA	151
10.2.28	HAL_CRYP_ErrorCallback.....	151
10.2.29	HAL_CRYP_InCpltCallback	151
10.2.30	HAL_CRYP_OutCpltCallback	152
10.2.31	HAL_CRYP_IRQHandler.....	152
10.2.32	HAL_CRYP_GetState	152
10.3	CRYP Firmware driver defines.....	153
10.3.1	CRYP	153
11	HAL CRYP Extension Driver.....	155
11.1	CRYPEx Firmware driver API description	155

11.1.1	Extended features functions	155
11.1.2	HAL_CRYPEx_ComputationCpltCallback.....	155
11.2	CRYPEx Firmware driver defines.....	155
11.2.1	CRYPEx	155
12	HAL DAC Generic Driver.....	156
12.1	DAC Firmware driver registers structures	156
12.1.1	DAC_HandleTypeDef	156
12.1.2	DAC_ChannelConfTypeDef	156
12.2	DAC Firmware driver API description.....	156
12.2.1	DAC Peripheral features.....	157
12.2.2	How to use this driver	158
12.2.3	Initialization and de-initialization functions	159
12.2.4	IO operation functions	159
12.2.5	Peripheral Control functions	159
12.2.6	Peripheral State and Errors functions	160
12.2.7	HAL_DAC_Init	160
12.2.8	HAL_DAC_DelInit.....	160
12.2.9	HAL_DAC_MspInit	161
12.2.10	HAL_DAC_MspDelInit.....	161
12.2.11	HAL_DAC_Start	161
12.2.12	HAL_DAC_Stop.....	162
12.2.13	HAL_DAC_Start_DMA	162
12.2.14	HAL_DAC_Stop_DMA.....	163
12.2.15	HAL_DAC_GetValue	163
12.2.16	HAL_DAC_IRQHandler	164
12.2.17	HAL_DAC_ConvCpltCallbackCh1.....	164
12.2.18	HAL_DAC_ConvHalfCpltCallbackCh1	164
12.2.19	HAL_DAC_ErrorCallbackCh1	165
12.2.20	HAL_DAC_DMAUnderrunCallbackCh1	165
12.2.21	HAL_DAC_SetValue	165
12.2.22	HAL_DAC_ConfigChannel	166
12.2.23	HAL_DAC_GetState.....	166
12.2.24	HAL_DAC_GetError	167
12.2.25	HAL_DAC_ConfigChannel	167
12.2.26	HAL_DAC_SetValue	167
12.2.27	HAL_DAC_GetState.....	168
12.2.28	HAL_DAC_GetError	168
12.3	DAC Firmware driver defines	169

12.3.1	DAC	169
13	HAL DAC Extension Driver	172
13.1	DACE Firmware driver API description	172
13.1.1	How to use this driver	172
13.1.2	Extended features functions	172
13.1.3	HAL_DACE_DualGetValue	172
13.1.4	HAL_DACE_TriangleWaveGenerate	173
13.1.5	HAL_DACE_NoiseWaveGenerate	174
13.1.6	HAL_DACE_DualSetValue	174
13.1.7	HAL_DACE_ConvCpltCallbackCh2	175
13.1.8	HAL_DACE_ConvHalfCpltCallbackCh2	175
13.1.9	HAL_DACE_ErrorCallbackCh2	176
13.1.10	HAL_DACE_DMAUnderrunCallbackCh2	176
13.2	DACE Firmware driver defines	177
13.2.1	DACE	177
14	HAL DMA Generic Driver	179
14.1	DMA Firmware driver registers structures	179
14.1.1	DMA_InitTypeDef	179
14.1.2	_DMA_HandleTypeDef	179
14.2	DMA Firmware driver API description	180
14.2.1	How to use this driver	180
14.2.2	Initialization and de-initialization functions	181
14.2.3	IO operation functions	181
14.2.4	State and Errors functions	182
14.2.5	HAL_DMA_Init	182
14.2.6	HAL_DMA_DeInit	182
14.2.7	HAL_DMA_Start	182
14.2.8	HAL_DMA_Start_IT	183
14.2.9	HAL_DMA_Abort	183
14.2.10	HAL_DMA_PollForTransfer	184
14.2.11	HAL_DMA_IRQHandler	184
14.2.12	HAL_DMA_GetState	185
14.2.13	HAL_DMA_GetError	185
14.3	DMA Firmware driver defines	185
14.3.1	DMA	185
15	HAL DMA Extension Driver	189
15.1	DMAEx Firmware driver defines	189

15.1.1	DMAEx.....	189
16	HAL FLASH Generic Driver.....	190
16.1	FLASH Firmware driver registers structures	190
16.1.1	FLASH_ProcessTypeDef	190
16.2	FLASH Firmware driver API description.....	190
16.2.1	FLASH peripheral features	190
16.2.2	How to use this driver	190
16.2.3	Programming operation functions	191
16.2.4	Option Bytes Programming functions.....	192
16.2.5	Peripheral Control functions	192
16.2.6	Peripheral Errors functions	192
16.2.7	HAL_FLASH_Program	193
16.2.8	HAL_FLASH_Program_IT	193
16.2.9	HAL_FLASH_EndOfOperationCallback	193
16.2.10	HAL_FLASH_OperationErrorHandler	194
16.2.11	HAL_FLASH_IRQHandler	194
16.2.12	HAL_FLASH_Unlock.....	195
16.2.13	HAL_FLASH_Lock	195
16.2.14	HAL_FLASH_OB_Unlock.....	195
16.2.15	HAL_FLASH_OB_Lock	195
16.2.16	HAL_FLASH_OB_Launch.....	196
16.2.17	HAL_FLASH_GetError	196
16.3	FLASH Firmware driver defines	197
16.3.1	FLASH	197
17	HAL FLASH Extension Driver.....	199
17.1	FLASHEx Firmware driver registers structures	199
17.1.1	FLASH_EraseInitTypeDef	199
17.1.2	FLASH_OBProgramInitTypeDef	199
17.1.3	FLASH_AdvOBProgramInitTypeDef	200
17.2	FLASHEx Firmware driver API description.....	200
17.2.1	Flash peripheral Extended features	200
17.2.2	How to use this driver	200
17.2.3	FLASH Erasing Programming functions.....	201
17.2.4	Option Bytes Programming functions.....	201
17.2.5	DATA EEPROM Programming functions	202
17.2.6	HAL_FLASHEx_Erase	202
17.2.7	HAL_FLASHEx_Erase_IT	202
17.2.8	HAL_FLASHEx_OBProgram.....	203

17.2.9	HAL_FLASHEx_OBGetConfig	203
17.2.10	HAL_FLASHEx_AdvOBProgram	204
17.2.11	HAL_FLASHEx_AdvOBGetConfig	204
17.2.12	HAL_FLASHEx_DATAEEPROM_Unlock	204
17.2.13	HAL_FLASHEx_DATAEEPROM_Lock.....	205
17.2.14	HAL_FLASHEx_DATAEEPROM_Erase	205
17.2.15	HAL_FLASHEx_DATAEEPROM_Program.....	205
17.2.16	HAL_FLASHEx_DATAEEPROM_EnableFixedTimeProgram	206
17.2.17	HAL_FLASHEx_DATAEEPROM_DisableFixedTimeProgram.....	206
17.3	FLASHEx Firmware driver defines	207
17.3.1	FLASHEx	207
18	HAL FLASH__RAMFUNC Generic Driver.....	212
18.1	FLASH__RAMFUNC Firmware driver API description	212
18.1.1	HAL_FLASHEx_EnableRunPowerDown	212
18.1.2	HAL_FLASHEx_DisableRunPowerDown.....	212
18.1.3	HAL_FLASHEx_EraseParallelPage.....	212
18.1.4	HAL_FLASHEx_ProgramParallelHalfPage	213
18.1.5	HAL_FLASHEx_HalfPageProgram	214
18.1.6	HAL_FLASHEx_DATAEEPROM_EraseDoubleWord	215
18.1.7	HAL_FLASHEx_DATAEEPROM_ProgramDoubleWord	215
19	HAL GPIO Generic Driver.....	217
19.1	GPIO Firmware driver registers structures	217
19.1.1	GPIO_InitTypeDef	217
19.2	GPIO Firmware driver API description	217
19.2.1	GPIO Peripheral features	217
19.2.2	How to use this driver	218
19.2.3	Initialization and de-initialization functions	218
19.2.4	HAL_GPIO_Init.....	219
19.2.5	HAL_GPIO_DelInit	219
19.2.6	HAL_GPIO_ReadPin.....	219
19.2.7	HAL_GPIO_WritePin.....	220
19.2.8	HAL_GPIO_TogglePin	220
19.2.9	HAL_GPIO_LockPin.....	221
19.2.10	HAL_GPIO_EXTI_IRQHandler	221
19.2.11	HAL_GPIO_EXTI_Callback.....	221
19.3	GPIO Firmware driver defines	222
19.3.1	GPIO	222

20 HAL GPIO Extension Driver.....	225
20.1 GPIOEx Firmware driver defines.....	225
20.1.1 GPIOEx	225
21 HAL I2C Generic Driver.....	227
21.1 I2C Firmware driver registers structures	227
21.1.1 I2C_InitTypeDef.....	227
21.1.2 I2C_HandleTypeDef	227
21.2 I2C Firmware driver API description.....	228
21.2.1 How to use this driver	228
21.2.2 Initialization and de-initialization functions	231
21.2.3 IO operation functions	231
21.2.4 Peripheral State and Errors functions	233
21.2.5 HAL_I2C_Init	233
21.2.6 HAL_I2C_DeInit.....	233
21.2.7 HAL_I2C_MspInit	233
21.2.8 HAL_I2C_MspDeInit.....	234
21.2.9 HAL_I2C_Master_Transmit.....	234
21.2.10 HAL_I2C_Master_Receive.....	235
21.2.11 HAL_I2C_Slave_Transmit.....	235
21.2.12 HAL_I2C_Slave_Receive.....	235
21.2.13 HAL_I2C_Master_Transmit_IT.....	236
21.2.14 HAL_I2C_Master_Receive_IT.....	236
21.2.15 HAL_I2C_Slave_Transmit_IT.....	237
21.2.16 HAL_I2C_Slave_Receive_IT.....	237
21.2.17 HAL_I2C_Master_Transmit_DMA.....	238
21.2.18 HAL_I2C_Master_Receive_DMA.....	238
21.2.19 HAL_I2C_Slave_Transmit_DMA.....	238
21.2.20 HAL_I2C_Slave_Receive_DMA.....	239
21.2.21 HAL_I2C_Mem_Write.....	239
21.2.22 HAL_I2C_Mem_Read	240
21.2.23 HAL_I2C_Mem_Write_IT	240
21.2.24 HAL_I2C_Mem_Read_IT	241
21.2.25 HAL_I2C_Mem_Write_DMA	241
21.2.26 HAL_I2C_Mem_Read_DMA	242
21.2.27 HAL_I2C_IsDeviceReady.....	242
21.2.28 HAL_I2C_EV_IRQHandler.....	243
21.2.29 HAL_I2C_ER_IRQHandler.....	243
21.2.30 HAL_I2C_MasterTxCpltCallback.....	243

21.2.31	HAL_I2C_MasterRxCpltCallback	244
21.2.32	HAL_I2C_SlaveTxCpltCallback.....	244
21.2.33	HAL_I2C_SlaveRxCpltCallback	244
21.2.34	HAL_I2C_MemTxCpltCallback.....	245
21.2.35	HAL_I2C_MemRxCpltCallback	245
21.2.36	HAL_I2C_ErrorCallback	245
21.2.37	HAL_I2C_GetState	246
21.2.38	HAL_I2C_GetError	246
21.3	I2C Firmware driver defines	247
21.3.1	I2C	247
22	HAL I2S Generic Driver	250
22.1	I2S Firmware driver registers structures	250
22.1.1	I2S_InitTypeDef.....	250
22.1.2	I2S_HandleTypeDef	250
22.2	I2S Firmware driver API description.....	251
22.2.1	How to use this driver	251
22.2.2	Initialization and de-initialization functions	253
22.2.3	IO operation functions	253
22.2.4	Peripheral State and Errors functions	254
22.2.5	HAL_I2S_Init	254
22.2.6	HAL_I2S_DeInit.....	254
22.2.7	HAL_I2S_MspInit.....	255
22.2.8	HAL_I2S_MspDeInit.....	255
22.2.9	HAL_I2S_Transmit	255
22.2.10	HAL_I2S_Receive	256
22.2.11	HAL_I2S_Transmit_IT	257
22.2.12	HAL_I2S_Receive_IT	257
22.2.13	HAL_I2S_Transmit_DMA	258
22.2.14	HAL_I2S_Receive_DMA	258
22.2.15	HAL_I2S_DMAPause	259
22.2.16	HAL_I2S_DMAResume.....	259
22.2.17	HAL_I2S_DMAStop.....	260
22.2.18	HAL_I2S_IRQHandler	260
22.2.19	HAL_I2S_TxHalfCpltCallback	260
22.2.20	HAL_I2S_TxCpltCallback	261
22.2.21	HAL_I2S_RxHalfCpltCallback	261
22.2.22	HAL_I2S_RxCpltCallback	261
22.2.23	HAL_I2S_ErrorCallback	262

22.2.24	HAL_I2S_GetState	262
22.2.25	HAL_I2S_GetError	262
22.3	I2S Firmware driver defines	263
22.3.1	I2S	263
23	HAL IRDA Generic Driver.....	266
23.1	IRDA Firmware driver registers structures	266
23.1.1	IRDA_InitTypeDef.....	266
23.1.2	IRDA_HandleTypeDef	266
23.2	IRDA Firmware driver API description.....	267
23.2.1	How to use this driver	267
23.2.2	Initialization and Configuration functions.....	269
23.2.3	IO operation functions	269
23.2.4	Peripheral State and Errors functions	270
23.2.5	HAL_IRDA_Init	271
23.2.6	HAL_IRDA_DeInit.....	271
23.2.7	HAL_IRDA_MspInit	271
23.2.8	HAL_IRDA_MspDeInit.....	272
23.2.9	HAL_IRDA_Transmit.....	272
23.2.10	HAL_IRDA_Receive	272
23.2.11	HAL_IRDA_Transmit_IT	273
23.2.12	HAL_IRDA_Receive_IT	273
23.2.13	HAL_IRDA_Transmit_DMA.....	274
23.2.14	HAL_IRDA_Receive_DMA	274
23.2.15	HAL_IRDA_DMAPause.....	275
23.2.16	HAL_IRDA_DMAResume.....	275
23.2.17	HAL_IRDA_DMAPause.....	275
23.2.18	HAL_IRDA_IRQHandler	276
23.2.19	HAL_IRDA_TxCpltCallback	276
23.2.20	HAL_IRDA_TxHalfCpltCallback	276
23.2.21	HAL_IRDA_RxCpltCallback	277
23.2.22	HAL_IRDA_RxHalfCpltCallback	277
23.2.23	HAL_IRDA_ErrorCallback	277
23.2.24	HAL_IRDA_GetState	278
23.2.25	HAL_IRDA_GetError	278
23.3	IRDA Firmware driver defines	279
23.3.1	IRDA	279
24	HAL IWDG Generic Driver.....	283
24.1	IWDG Firmware driver registers structures	283

24.1.1	IWDG_InitTypeDef	283
24.1.2	IWDG_HandleTypeDef.....	283
24.2	IWDG Firmware driver API description	283
24.2.1	Initialization and de-initialization functions	283
24.2.2	IO operation functions	284
24.2.3	Peripheral State functions	284
24.2.4	HAL_IWDG_Init.....	284
24.2.5	HAL_IWDG_MspInit	284
24.2.6	HAL_IWDG_Start	285
24.2.7	HAL_IWDG_Refresh	285
24.2.8	HAL_IWDG_GetState.....	285
24.3	IWDG Firmware driver defines	286
24.3.1	IWDG	286
25	HAL LCD Generic Driver	288
25.1	LCD Firmware driver registers structures.....	288
25.1.1	LCD_InitTypeDef	288
25.1.2	LCD_HandleTypeDef	288
25.2	LCD Firmware driver API description	289
25.2.1	How to use this driver	289
25.2.2	HAL_LCD_DelInit	290
25.2.3	HAL_LCD_Init.....	290
25.2.4	HAL_LCD_MspInit.....	291
25.2.5	HAL_LCD_MspDeInit	291
25.2.6	HAL_LCD_Write	291
25.2.7	HAL_LCD_Clear	292
25.2.8	HAL_LCD_UpdateDisplayRequest	292
25.2.9	HAL_LCD_GetState	292
25.2.10	HAL_LCD_GetError.....	292
25.3	LCD Firmware driver defines.....	293
25.3.1	LCD.....	293
26	HAL NOR Generic Driver.....	300
26.1	NOR Firmware driver registers structures.....	300
26.1.1	NOR_IDTypeDef	300
26.1.2	NOR_CFITTypeDef	300
26.1.3	NOR_HandleTypeDef.....	300
26.2	NOR Firmware driver API description	301
26.2.1	How to use this driver	301

26.2.2	NOR Initialization and de_initialization functions	302
26.2.3	NOR Input and Output functions	302
26.2.4	NOR Control functions.....	302
26.2.5	NOR State functions.....	302
26.2.6	HAL_NOR_Init.....	302
26.2.7	HAL_NOR_Delnit	303
26.2.8	HAL_NOR_MsplInit.....	303
26.2.9	HAL_NOR_MspDelnit	304
26.2.10	HAL_NOR_MspWait.....	304
26.2.11	HAL_NOR_Read_ID	304
26.2.12	HAL_NOR_ReturnToReadMode.....	305
26.2.13	HAL_NOR_Read	305
26.2.14	HAL_NOR_Program.....	305
26.2.15	HAL_NOR_ReadBuffer	306
26.2.16	HAL_NOR_ProgramBuffer	306
26.2.17	HAL_NOR_Erase_Block	307
26.2.18	HAL_NOR_Erase_Chip.....	307
26.2.19	HAL_NOR_Read_CFI	307
26.2.20	HAL_NOR_WriteOperation_Enable	308
26.2.21	HAL_NOR_WriteOperation_Disable	308
26.2.22	HAL_NOR_GetState	309
26.2.23	HAL_NOR_GetStatus.....	309
26.3	NOR Firmware driver defines.....	309
26.3.1	NOR.....	309
27	HAL OPAMP Generic Driver	311
27.1	OPAMP Firmware driver registers structures	311
27.1.1	OPAMP_InitTypeDef	311
27.1.2	OPAMP_HandleTypeDef.....	312
27.2	OPAMP Firmware driver API description	312
27.2.1	OPAMP Peripheral Features	312
27.2.2	How to use this driver	313
27.2.3	Initialization and de-initialization functions	314
27.2.4	IO operation functions	314
27.2.5	Peripheral Control functions	314
27.2.6	Peripheral State functions	314
27.2.7	HAL_OPAMP_Init.....	315
27.2.8	HAL_OPAMP_Delnit	315
27.2.9	HAL_OPAMP_MsplInit.....	315

27.2.10	HAL_OPAMP_MspDeInit	316
27.2.11	HAL_OPAMP_GetTrimOffset.....	316
27.2.12	HAL_OPAMP_Lock.....	317
27.2.13	HAL_OPAMP_GetState	317
27.3	OPAMP Firmware driver defines.....	317
27.3.1	OPAMP	317
28	HAL OPAMP Extension Driver.....	321
28.1	OPAMPEx Firmware driver API description	321
28.1.1	Peripheral Control functions	321
28.1.2	HAL_OPAMPEx_SelfCalibrateAll.....	321
28.1.3	HAL_OPAMPEx_Unlock	321
28.2	OPAMPEx Firmware driver defines.....	322
28.2.1	OPAMPEx	322
29	HAL PCD Generic Driver	323
29.1	PCD Firmware driver registers structures	323
29.1.1	PCD_InitTypeDef.....	323
29.1.2	PCD_EPTTypeDef.....	323
29.1.3	PCD_HandleTypeDef.....	324
29.2	PCD Firmware driver API description.....	325
29.2.1	How to use this driver	325
29.2.2	Initialization and de-initialization functions	325
29.2.3	IO operation functions	325
29.2.4	Peripheral Control functions	326
29.2.5	Peripheral State functions	326
29.2.6	HAL_PCD_Init	326
29.2.7	HAL_PCD_DeInit.....	326
29.2.8	HAL_PCD_MspInit	327
29.2.9	HAL_PCD_MspDeInit.....	327
29.2.10	HAL_PCD_Start	327
29.2.11	HAL_PCD_Stop.....	328
29.2.12	HAL_PCD_IRQHandler	328
29.2.13	HAL_PCD_DataOutStageCallback	328
29.2.14	HAL_PCD_DataInStageCallback	329
29.2.15	HAL_PCD_SetupStageCallback	329
29.2.16	HAL_PCD_SOFCallback.....	329
29.2.17	HAL_PCD_ResetCallback.....	330
29.2.18	HAL_PCD_SuspendCallback.....	330

29.2.19	HAL_PCD_ResumeCallback.....	330
29.2.20	HAL_PCD_ISOOUTIncompleteCallback.....	331
29.2.21	HAL_PCD_ISOINIncompleteCallback.....	331
29.2.22	HAL_PCD_ConnectCallback.....	331
29.2.23	HAL_PCD_DisconnectCallback	332
29.2.24	HAL_PCD_DevConnect	332
29.2.25	HAL_PCD_DevDisconnect.....	332
29.2.26	HAL_PCD_SetAddress	333
29.2.27	HAL_PCD_EP_Open	333
29.2.28	HAL_PCD_EP_Close	334
29.2.29	HAL_PCD_EP_Receive	334
29.2.30	HAL_PCD_EP_GetRxCount	334
29.2.31	HAL_PCD_EP_Transmit	335
29.2.32	HAL_PCD_EP_SetStall.....	335
29.2.33	HAL_PCD_EP_ClrStall.....	335
29.2.34	HAL_PCD_EP_Flush	336
29.2.35	HAL_PCD_ActiveRemoteWakeup	336
29.2.36	HAL_PCD_DeActiveRemoteWakeup.....	336
29.2.37	HAL_PCD_GetState.....	337
29.2.38	HAL_PCDEx_SetConnectionState.....	337
29.3	PCD Firmware driver defines	338
29.3.1	PCD	338
30	HAL PCD Extension Driver	343
30.1	PCDEx Firmware driver API description	343
30.1.1	Peripheral Control functions	343
30.1.2	HAL_PCDEx_PMAConfig	343
30.2	PCDEx Firmware driver defines	343
30.2.1	PCDEx	343
31	HAL PWR Generic Driver	344
31.1	PWR Firmware driver registers structures	344
31.1.1	PWR_PVDTTypeDef	344
31.2	PWR Firmware driver API description.....	344
31.2.1	Initialization and de-initialization functions	344
31.2.2	Peripheral Control functions	344
31.2.3	HAL_PWR_DelInit.....	348
31.2.4	HAL_PWR_EnableBkUpAccess	348
31.2.5	HAL_PWR_DisableBkUpAccess.....	348
31.2.6	HAL_PWR_PVDConfig	349

31.2.7	HAL_PWR_EnablePVD.....	349
31.2.8	HAL_PWR_DisablePVD.....	349
31.2.9	HAL_PWR_EnableWakeUpPin.....	350
31.2.10	HAL_PWR_DisableWakeUpPin.....	350
31.2.11	HAL_PWR_EnterSLEEPMode.....	351
31.2.12	HAL_PWR_EnterSTOPMode.....	351
31.2.13	HAL_PWR_EnterSTANDBYMode	352
31.2.14	HAL_PWR_PVD_IRQHandler.....	352
31.2.15	HAL_PWR_PVDCallback.....	352
31.3	PWR Firmware driver defines	353
31.3.1	PWR	353
32	HAL PWR Extension Driver	356
32.1	PWREx Firmware driver API description.....	356
32.1.1	Peripheral extended features functions.....	356
32.1.2	HAL_PWREx_EnableFastWakeUp.....	356
32.1.3	HAL_PWREx_DisableFastWakeUp	356
32.1.4	HAL_PWREx_EnableUltraLowPower	356
32.1.5	HAL_PWREx_DisableUltraLowPower	357
32.1.6	HAL_PWREx_EnableLowPowerRunMode	357
32.1.7	HAL_PWREx_DisableLowPowerRunMode	357
32.2	PWREx Firmware driver defines	358
32.2.1	PWREx	358
33	HAL RCC Generic Driver.....	359
33.1	RCC Firmware driver registers structures	359
33.1.1	RCC_PLLInitTypeDef	359
33.1.2	RCC_OscInitTypeDef	359
33.1.3	RCC_ClkInitTypeDef	360
33.2	RCC Firmware driver API description	360
33.2.1	RCC specific features	360
33.2.2	Initialization and de-initialization functions	361
33.2.3	Peripheral Control functions	362
33.2.4	HAL_RCC_DeInit	363
33.2.5	HAL_RCC_OscConfig	363
33.2.6	HAL_RCC_ClockConfig	363
33.2.7	HAL_RCC_MCOConfig	364
33.2.8	HAL_RCC_EnableCSS	365
33.2.9	HAL_RCC_DisableCSS	365

33.2.10	HAL_RCC_GetSysClockFreq	366
33.2.11	HAL_RCC_GetHCLKFreq	366
33.2.12	HAL_RCC_GetPCLK1Freq	367
33.2.13	HAL_RCC_GetPCLK2Freq	367
33.2.14	HAL_RCC_GetOscConfig	367
33.2.15	HAL_RCC_GetClockConfig	368
33.2.16	HAL_RCC_NMI_IRQHandler	368
33.2.17	HAL_RCC_CCSCallback	368
33.3	RCC Firmware driver defines	369
33.3.1	RCC	369
34	HAL RCC Extension Driver	380
34.1	RCCEEx Firmware driver registers structures	380
34.1.1	RCC_PeriphCLKInitTypeDef	380
34.2	RCCEEx Firmware driver API description	380
34.2.1	Extended Peripheral Control functions	380
34.2.2	HAL_RCCEEx_PeriphCLKConfig	380
34.2.3	HAL_RCCEEx_GetPeriphCLKConfig	381
34.2.4	HAL_RCCEEx_EnableLSECSS	381
34.2.5	HAL_RCCEEx_DisableLSECSS	382
34.3	RCCEEx Firmware driver defines	382
34.3.1	RCCEx	382
35	HAL RTC Generic Driver	385
35.1	RTC Firmware driver registers structures	385
35.1.1	RTC_InitTypeDef	385
35.1.2	RTC_DateTypeDef	385
35.1.3	RTC_HandleTypeDef	386
35.2	RTC Firmware driver API description	386
35.2.1	Backup Domain Operating Condition	386
35.2.2	Backup Domain Reset	387
35.2.3	Backup Domain Access	387
35.2.4	How to use this driver	387
35.2.5	RTC and low power modes	387
35.2.6	Initialization and de-initialization functions	388
35.2.7	RTC Time and Date functions	388
35.2.8	RTC Alarm functions	389
35.2.9	Peripheral State functions	389
35.2.10	Peripheral Control functions	389

35.2.11	HAL_RTC_Init	389
35.2.12	HAL_RTC_DelInit.....	390
35.2.13	HAL_RTC_MspInit.....	390
35.2.14	HAL_RTC_MspDeInit.....	390
35.2.15	HAL_RTC_SetTime.....	391
35.2.16	HAL_RTC_GetTime	391
35.2.17	HAL_RTC_SetDate	392
35.2.18	HAL_RTC_GetDate	392
35.2.19	HAL_RTC_SetTime.....	393
35.2.20	HAL_RTC_SetDate	393
35.2.21	HAL_RTC_GetDate	393
35.2.22	HAL_RTC_GetTime	394
35.2.23	HAL_RTC_SetAlarm	394
35.2.24	HAL_RTC_SetAlarm_IT	395
35.2.25	HAL_RTC_DeactivateAlarm.....	395
35.2.26	HAL_RTC_GetAlarm.....	396
35.2.27	HAL_RTC_AlarmIRQHandler.....	396
35.2.28	HAL_RTC_PollForAlarmAEvent.....	397
35.2.29	HAL_RTC_AlarmAEventCallback	397
35.2.30	HAL_RTC_DeactivateAlarm.....	397
35.2.31	HAL_RTC_AlarmIRQHandler.....	398
35.2.32	HAL_RTC_AlarmAEventCallback	398
35.2.33	HAL_RTC_PollForAlarmAEvent.....	399
35.2.34	HAL_RTC_SetAlarm	399
35.2.35	HAL_RTC_SetAlarm_IT	399
35.2.36	HAL_RTC_GetAlarm	400
35.2.37	HAL_RTC_WaitForSynchro	400
35.2.38	HAL_RTC_GetState	401
35.2.39	HAL_RTC_WaitForSynchro	401
35.3	RTC Firmware driver defines	402
35.3.1	RTC	402
36	HAL RTC Extension Driver	408
36.1	RTCEx Firmware driver registers structures	408
36.1.1	RTC_TamperTypeDef	408
36.1.2	RTC_TimeTypeDef.....	408
36.1.3	RTC_AlarmTypeDef	409
36.2	RTCEx Firmware driver API description.....	410
36.2.1	How to use this driver	410

36.2.2	RTCTimeStamp and Tamper functions	410
36.2.3	RTC Wake-up functions	411
36.2.4	Extension Peripheral Control functions	411
36.2.5	Extended features functions	412
36.2.6	HAL_RTCEx_SetTimeStamp	412
36.2.7	HAL_RTCEx_SetTimeStamp_IT	412
36.2.8	HAL_RTCEx_DeactivateTimeStamp	413
36.2.9	HAL_RTCEx_GetTimeStamp.....	413
36.2.10	HAL_RTCEx_SetTamper	414
36.2.11	HAL_RTCEx_SetTamper_IT	414
36.2.12	HAL_RTCEx_DeactivateTamper	414
36.2.13	HAL_RTCEx_TamperTimeStampIRQHandler	415
36.2.14	HAL_RTCEx_TimeStampEventCallback	415
36.2.15	HAL_RTCEx_Tamper1EventCallback	416
36.2.16	HAL_RTCEx_Tamper2EventCallback	416
36.2.17	HAL_RTCEx_Tamper3EventCallback	416
36.2.18	HAL_RTCEx_PollForTimeStampEvent.....	417
36.2.19	HAL_RTCEx_PollForTamper1Event.....	417
36.2.20	HAL_RTCEx_PollForTamper2Event.....	417
36.2.21	HAL_RTCEx_PollForTamper3Event.....	418
36.2.22	HAL_RTCEx_SetWakeUpTimer	418
36.2.23	HAL_RTCEx_SetWakeUpTimer_IT	418
36.2.24	HAL_RTCEx_DeactivateWakeUpTimer.....	419
36.2.25	HAL_RTCEx_GetWakeUpTimer	419
36.2.26	HAL_RTCEx_WakeUpTimerIRQHandler.....	420
36.2.27	HAL_RTCEx_WakeUpTimerEventCallback	420
36.2.28	HAL_RTCEx_PollForWakeUpTimerEvent	420
36.2.29	HAL_RTCEx_BKUPWrite.....	421
36.2.30	HAL_RTCEx_BKUPRead	421
36.2.31	HAL_RTCEx_SetCoarseCalib.....	421
36.2.32	HAL_RTCEx_DeactivateCoarseCalib	422
36.2.33	HAL_RTCEx_SetSmoothCalib.....	422
36.2.34	HAL_RTCEx_SetSynchroShift	423
36.2.35	HAL_RTCEx_SetCalibrationOutPut	424
36.2.36	HAL_RTCEx_DeactivateCalibrationOutPut	424
36.2.37	HAL_RTCEx_SetRefClock.....	425
36.2.38	HAL_RTCEx_DeactivateRefClock	425
36.2.39	HAL_RTCEx_EnableBypassShadow	425
36.2.40	HAL_RTCEx_DisableBypassShadow	426

36.2.41	HAL_RTCEx_AlarmBEventCallback	426
36.2.42	HAL_RTCEx_PollForAlarmBEvent	426
36.3	RTCEx Firmware driver defines	427
36.3.1	RTCEx	427
37	HAL SD Generic Driver	433
37.1	SD Firmware driver registers structures	433
37.1.1	SD_HandleTypeDef.....	433
37.1.2	HAL_SD_CSDTypeDef.....	433
37.1.3	HAL_SD_CIDTypeDef	435
37.1.4	HAL_SD_CardStatusTypeDef	436
37.1.5	HAL_SD_CardInfoTypeDef.....	436
37.2	SD Firmware driver API description	437
37.2.1	How to use this driver	437
37.2.2	Initialization and de-initialization functions	439
37.2.3	IO operation functions	439
37.2.4	Peripheral Control functions	440
37.2.5	Peripheral State functions	440
37.2.6	HAL_SD_Init.....	440
37.2.7	HAL_SD_DelInit	440
37.2.8	HAL_SD_MspInit	441
37.2.9	HAL_SD_MspDelInit	441
37.2.10	HAL_SD_ReadBlocks	441
37.2.11	HAL_SD_WriteBlocks.....	442
37.2.12	HAL_SD_ReadBlocks_DMA	442
37.2.13	HAL_SD_WriteBlocks_DMA	443
37.2.14	HAL_SD_CheckReadOperation.....	443
37.2.15	HAL_SD_CheckWriteOperation	444
37.2.16	HAL_SD_Erase	444
37.2.17	HAL_SD_IRQHandler.....	444
37.2.18	HAL_SD_XferCpltCallback.....	445
37.2.19	HAL_SD_XferErrorCallback	445
37.2.20	HAL_SD_DMA_RxCpltCallback.....	445
37.2.21	HAL_SD_DMA_RxErrorCallback	446
37.2.22	HAL_SD_DMA_TxCpltCallback	446
37.2.23	HAL_SD_DMA_TxErrorCallback.....	446
37.2.24	HAL_SD_Get_CardInfo.....	447
37.2.25	HAL_SD_WideBusOperation_Config.....	447
37.2.26	HAL_SD_StopTransfer.....	448

37.2.27	HAL_SD_HighSpeed.....	448
37.2.28	HAL_SD_SendSDStatus.....	448
37.2.29	HAL_SD_GetStatus.....	449
37.2.30	HAL_SD_GetCardStatus.....	449
37.3	SD Firmware driver defines	449
37.3.1	SD	449
38	HAL SMARTCARD Generic Driver.....	457
38.1	SMARTCARD Firmware driver registers structures	457
38.1.1	SMARTCARD_InitTypeDef	457
38.1.2	SMARTCARD_HandleTypeDef.....	458
38.2	SMARTCARD Firmware driver API description.....	459
38.2.1	How to use this driver	459
38.2.2	Initialization and Configuration functions.....	460
38.2.3	IO operation functions	461
38.2.4	Peripheral State and Errors functions	463
38.2.5	HAL_SMARTCARD_Init	463
38.2.6	HAL_SMARTCARD_DelInit	463
38.2.7	HAL_SMARTCARD_MspInit	464
38.2.8	HAL_SMARTCARD_MspDelInit	464
38.2.9	HAL_SMARTCARD_Transmit.....	465
38.2.10	HAL_SMARTCARD_Receive.....	465
38.2.11	HAL_SMARTCARD_Transmit_IT	465
38.2.12	HAL_SMARTCARD_Receive_IT	466
38.2.13	HAL_SMARTCARD_Transmit_DMA.....	466
38.2.14	HAL_SMARTCARD_Receive_DMA.....	467
38.2.15	HAL_SMARTCARD_IRQHandler.....	467
38.2.16	HAL_SMARTCARD_TxCpltCallback	467
38.2.17	HAL_SMARTCARD_RxCpltCallback	468
38.2.18	HAL_SMARTCARD_ErrorCallback	468
38.2.19	HAL_SMARTCARD_GetState	469
38.2.20	HAL_SMARTCARD_GetError	469
38.3	SMARTCARD Firmware driver defines	469
38.3.1	SMARTCARD	469
39	HAL SPI Generic Driver.....	474
39.1	SPI Firmware driver registers structures	474
39.1.1	SPI_InitTypeDef	474
39.1.2	__SPI_HandleTypeDef.....	475

39.2	SPI Firmware driver API description	475
39.2.1	How to use this driver	475
39.2.2	Initialization and de-initialization functions	476
39.2.3	IO operation functions	477
39.2.4	Peripheral State and Errors functions	478
39.2.5	HAL_SPI_Init	478
39.2.6	HAL_SPI_Delnit	478
39.2.7	HAL_SPI_MspInit	479
39.2.8	HAL_SPI_MspDeInit.....	479
39.2.9	HAL_SPI_Transmit.....	479
39.2.10	HAL_SPI_Receive.....	480
39.2.11	HAL_SPI_TransmitReceive.....	480
39.2.12	HAL_SPI_Transmit_IT.....	481
39.2.13	HAL_SPI_Receive_IT.....	481
39.2.14	HAL_SPI_TransmitReceive_IT	481
39.2.15	HAL_SPI_Transmit_DMA.....	482
39.2.16	HAL_SPI_Receive_DMA.....	482
39.2.17	HAL_SPI_TransmitReceive_DMA.....	483
39.2.18	HAL_SPI_DMAPause.....	483
39.2.19	HAL_SPI_DMAResume	483
39.2.20	HAL_SPI_DMAStop	484
39.2.21	HAL_SPI_IRQHandler.....	484
39.2.22	HAL_SPI_TxCpltCallback	484
39.2.23	HAL_SPI_RxCpltCallback	485
39.2.24	HAL_SPI_TxRxCpltCallback	485
39.2.25	HAL_SPI_TxHalfCpltCallback	485
39.2.26	HAL_SPI_RxHalfCpltCallback	486
39.2.27	HAL_SPI_TxRxBalCpltCallback	486
39.2.28	HAL_SPI_ErrorCallback	487
39.2.29	HAL_SPI_GetState.....	487
39.2.30	HAL_SPI_GetError	487
39.3	SPI Firmware driver defines	488
39.3.1	SPI	488
40	HAL SRAM Generic Driver	492
40.1	SRAM Firmware driver registers structures.....	492
40.1.1	SRAM_HandleTypeDef	492
40.2	SRAM Firmware driver API description	492
40.2.1	How to use this driver	492

40.2.2	SRAM Initialization and de_initialization functions	493
40.2.3	SRAM Input and Output functions	493
40.2.4	SRAM Control functions	493
40.2.5	SRAM State functions	494
40.2.6	HAL_SRAM_Init	494
40.2.7	HAL_SRAM_DelInit.....	494
40.2.8	HAL_SRAM_MspInit.....	495
40.2.9	HAL_SRAM_MspDeInit.....	495
40.2.10	HAL_SRAM_DMA_XferCpltCallback	495
40.2.11	HAL_SRAM_DMA_XferErrorCallback.....	496
40.2.12	HAL_SRAM_Read_8b.....	496
40.2.13	HAL_SRAM_Write_8b.....	496
40.2.14	HAL_SRAM_Read_16b.....	497
40.2.15	HAL_SRAM_Write_16b.....	497
40.2.16	HAL_SRAM_Read_32b.....	498
40.2.17	HAL_SRAM_Write_32b.....	498
40.2.18	HAL_SRAM_Read_DMA.....	498
40.2.19	HAL_SRAM_Write_DMA.....	499
40.2.20	HAL_SRAM_WriteOperation_Enable.....	499
40.2.21	HAL_SRAM_WriteOperation_Disable.....	500
40.2.22	HAL_SRAM_GetState	500
40.3	SRAM Firmware driver defines	500
40.3.1	SRAM	500
41	HAL TIM Generic Driver	501
41.1	TIM Firmware driver registers structures.....	501
41.1.1	TIM_Base_InitTypeDef.....	501
41.1.2	TIM_OC_InitTypeDef.....	501
41.1.3	TIM_OnePulse_InitTypeDef	502
41.1.4	TIM_IC_InitTypeDef	502
41.1.5	TIM_Encoder_InitTypeDef	503
41.1.6	TIM_ClockConfigTypeDef	503
41.1.7	TIM_ClearInputConfigTypeDef.....	504
41.1.8	TIM_SlaveConfigTypeDef	504
41.1.9	TIM_HandleTypeDef	505
41.2	TIM Firmware driver API description	505
41.2.1	TIMER Generic features.....	505
41.2.2	How to use this driver.....	506
41.2.3	Time Base functions	507

41.2.4	Time Output Compare functions	507
41.2.5	Time PWM functions	507
41.2.6	Time Input Capture functions	508
41.2.7	Time One Pulse functions	508
41.2.8	Time Encoder functions.....	509
41.2.9	IRQ handler management	509
41.2.10	Peripheral Control functions	509
41.2.11	HAL_TIM_Base_Init	510
41.2.12	Peripheral State functions	510
41.2.13	HAL_TIM_Base_DelInit.....	510
41.2.14	HAL_TIM_Base_MspInit.....	511
41.2.15	HAL_TIM_Base_MspDelInit.....	511
41.2.16	HAL_TIM_Base_Start.....	511
41.2.17	HAL_TIM_Base_Stop.....	512
41.2.18	HAL_TIM_Base_Start_IT	512
41.2.19	HAL_TIM_Base_Stop_IT.....	512
41.2.20	HAL_TIM_Base_Start_DMA	513
41.2.21	HAL_TIM_Base_Stop_DMA.....	513
41.2.22	HAL_TIM_OC_Init	514
41.2.23	HAL_TIM_OC_DelInit.....	514
41.2.24	HAL_TIM_OC_MspInit	514
41.2.25	HAL_TIM_OC_MspDelInit.....	515
41.2.26	HAL_TIM_OC_Start	515
41.2.27	HAL_TIM_OC_Stop	515
41.2.28	HAL_TIM_OC_Start_IT	515
41.2.29	HAL_TIM_OC_Stop_IT	516
41.2.30	HAL_TIM_OC_Start_DMA	516
41.2.31	HAL_TIM_OC_Stop_DMA	517
41.2.32	HAL_TIM_PWM_Init.....	517
41.2.33	HAL_TIM_PWM_DelInit	518
41.2.34	HAL_TIM_PWM_MspInit	518
41.2.35	HAL_TIM_PWM_MspDelInit	518
41.2.36	HAL_TIM_PWM_Start.....	519
41.2.37	HAL_TIM_PWM_Stop	519
41.2.38	HAL_TIM_PWM_Start_IT	520
41.2.39	HAL_TIM_PWM_Stop_IT	520
41.2.40	HAL_TIM_PWM_Start_DMA	520
41.2.41	HAL_TIM_PWM_Stop_IT	521

41.2.42	HAL_TIM_PWM_Stop_DMA	521
41.2.43	HAL_TIM_IC_Init	522
41.2.44	HAL_TIM_IC_DelInit	522
41.2.45	HAL_TIM_IC_MspInit	523
41.2.46	HAL_TIM_IC_MspDelInit	523
41.2.47	HAL_TIM_IC_Start	523
41.2.48	HAL_TIM_IC_Stop	524
41.2.49	HAL_TIM_IC_Start_IT	524
41.2.50	HAL_TIM_IC_Stop_IT	524
41.2.51	HAL_TIM_IC_Start_DMA	525
41.2.52	HAL_TIM_IC_Stop_DMA	525
41.2.53	HAL_TIM_OnePulse_Init	526
41.2.54	HAL_TIM_OnePulse_DelInit	526
41.2.55	HAL_TIM_OnePulse_MspInit	527
41.2.56	HAL_TIM_OnePulse_MspDelInit	527
41.2.57	HAL_TIM_OnePulse_Start	527
41.2.58	HAL_TIM_OnePulse_Stop	528
41.2.59	HAL_TIM_OnePulse_Start_IT	528
41.2.60	HAL_TIM_OnePulse_Stop_IT	529
41.2.61	HAL_TIM_Encoder_Init	529
41.2.62	HAL_TIM_Encoder_DelInit	529
41.2.63	HAL_TIM_Encoder_MspInit	530
41.2.64	HAL_TIM_Encoder_MspDelInit	530
41.2.65	HAL_TIM_Encoder_Start	530
41.2.66	HAL_TIM_Encoder_Stop	531
41.2.67	HAL_TIM_Encoder_Start_IT	531
41.2.68	HAL_TIM_Encoder_Stop_IT	532
41.2.69	HAL_TIM_Encoder_Start_DMA	532
41.2.70	HAL_TIM_Encoder_Stop_DMA	533
41.2.71	HAL_TIM_IRQHandler	533
41.2.72	HAL_TIM_OC_ConfigChannel	533
41.2.73	HAL_TIM_IC_ConfigChannel	534
41.2.74	HAL_TIM_PWM_ConfigChannel	534
41.2.75	HAL_TIM_OnePulse_ConfigChannel	535
41.2.76	HAL_TIM_DMABurst_WriteStart	535
41.2.77	HAL_TIM_DMABurst_WriteStop	536
41.2.78	HAL_TIM_DMABurst_ReadStart	537
41.2.79	HAL_TIM_DMABurst_ReadStop	538
41.2.80	HAL_TIM_GenerateEvent	538

41.2.81	HAL_TIM_ConfigOCrefClear	539
41.2.82	HAL_TIM_ConfigClockSource	539
41.2.83	HAL_TIM_ConfigTI1Input	539
41.2.84	HAL_TIM_SlaveConfigSynchronization	540
41.2.85	HAL_TIM_ReadCapturedValue	540
41.2.86	HAL_TIM_PeriodElapsedCallback	541
41.2.87	HAL_TIM_OC_DelayElapsedCallback	541
41.2.88	HAL_TIM_IC_CaptureCallback	542
41.2.89	HAL_TIM_PWM_PulseFinishedCallback	542
41.2.90	HAL_TIM_TriggerCallback	542
41.2.91	HAL_TIM_ErrorCallback	543
41.2.92	HAL_TIM_Base_GetState	543
41.2.93	HAL_TIM_OC_GetState	543
41.2.94	HAL_TIM_PWM_GetState	544
41.2.95	HAL_TIM_IC_GetState	544
41.2.96	HAL_TIM_OnePulse_GetState	544
41.2.97	HAL_TIM_Encoder_GetState	545
41.3	TIM Firmware driver defines	545
41.3.1	TIM	545
42	HAL TIM Extension Driver	555
42.1	TIMEx Firmware driver registers structures	555
42.1.1	TIM_MasterConfigTypeDef	555
42.2	TIMEx Firmware driver API description	555
42.2.1	TIMER Extended features	555
42.2.2	Peripheral Control functions	555
42.2.3	HAL_TIMEx_MasterConfigSynchronization	555
42.2.4	HAL_TIMEx_RemapConfig	556
42.3	TIMEx Firmware driver defines	557
42.3.1	TIMEx	557
43	HAL UART Generic Driver	559
43.1	UART Firmware driver registers structures	559
43.1.1	UART_InitTypeDef	559
43.1.2	UART_HandleTypeDef	559
43.2	UART Firmware driver API description	560
43.2.1	How to use this driver	560
43.2.2	Initialization and Configuration functions	562
43.2.3	IO operation functions	563

43.2.4	Peripheral Control functions	564
43.2.5	Peripheral State and Errors functions	564
43.2.6	HAL_UART_Init	565
43.2.7	HAL_HalfDuplex_Init	565
43.2.8	HAL_LIN_Init	566
43.2.9	HAL_MultiProcessor_Init	566
43.2.10	HAL_UART_DelInit	567
43.2.11	HAL_UART_MspInit	567
43.2.12	HAL_UART_MspDelInit	567
43.2.13	HAL_UART_Transmit	568
43.2.14	HAL_UART_Receive	568
43.2.15	HAL_UART_Transmit_IT	569
43.2.16	HAL_UART_Receive_IT	569
43.2.17	HAL_UART_Transmit_DMA	569
43.2.18	HAL_UART_Receive_DMA	570
43.2.19	HAL_UART_DMAPause	570
43.2.20	HAL_UART_DMAResume	571
43.2.21	HAL_UART_DMAStop	571
43.2.22	HAL_UART_IRQHandler	571
43.2.23	HAL_UART_TxCpltCallback	572
43.2.24	HAL_UART_TxHalfCpltCallback	572
43.2.25	HAL_UART_RxCpltCallback	572
43.2.26	HAL_UART_RxHalfCpltCallback	573
43.2.27	HAL_UART_ErrorCallback	573
43.2.28	HAL_LIN_SendBreak	573
43.2.29	HAL_MultiProcessor_EnterMuteMode	574
43.2.30	HAL_MultiProcessor_ExitMuteMode	574
43.2.31	HAL_HalfDuplex_EnableTransmitter	575
43.2.32	HAL_HalfDuplex_EnableReceiver	575
43.2.33	HAL_UART_GetState	575
43.2.34	HAL_UART_GetError	576
43.3	UART Firmware driver defines	576
43.3.1	UART	576
44	HAL USART Generic Driver	581
44.1	USART Firmware driver registers structures	581
44.1.1	USART_InitTypeDef	581
44.1.2	USART_HandleTypeDef	581
44.2	USART Firmware driver API description	582

44.2.1	How to use this driver	582
44.2.2	Initialization and Configuration functions.....	584
44.2.3	IO operation functions	585
44.2.4	Peripheral State and Errors functions	586
44.2.5	HAL_USART_Init.....	586
44.2.6	HAL_USART_DeInit.....	587
44.2.7	HAL_USART_MspInit.....	587
44.2.8	HAL_USART_MspDeInit	587
44.2.9	HAL_USART_Transmit	588
44.2.10	HAL_USART_Receive	588
44.2.11	HAL_USART_TransmitReceive	589
44.2.12	HAL_USART_Transmit_IT	589
44.2.13	HAL_USART_Receive_IT	590
44.2.14	HAL_USART_TransmitReceive_IT	590
44.2.15	HAL_USART_Transmit_DMA	591
44.2.16	HAL_USART_Receive_DMA	591
44.2.17	HAL_USART_TransmitReceive_DMA	591
44.2.18	HAL_USART_DMAPause	592
44.2.19	HAL_USART_DMAResume	592
44.2.20	HAL_USART_DMAStop	593
44.2.21	HAL_USART_IRQHandler	593
44.2.22	HAL_USART_TxCpltCallback	593
44.2.23	HAL_USART_TxHalfCpltCallback	594
44.2.24	HAL_USART_RxCpltCallback	594
44.2.25	HAL_USART_RxHalfCpltCallback	595
44.2.26	HAL_USART_TxRxCpltCallback	595
44.2.27	HAL_USART_ErrorCallback	595
44.2.28	HAL_USART_GetState	596
44.2.29	HAL_USART_GetError.....	596
44.3	USART Firmware driver defines.....	596
44.3.1	USART.....	596
45	HAL WWDG Generic Driver	601
45.1	WWDG Firmware driver registers structures.....	601
45.1.1	WWDG_InitTypeDef	601
45.1.2	WWDG_HandleTypeDef	601
45.2	WWDG Firmware driver API description	601
45.2.1	Initialization and de-initialization functions	601
45.2.2	IO operation functions	602

Contents	UM1816
45.2.3 Peripheral State functions	602
45.2.4 HAL_WWDG_Init.....	602
45.2.5 HAL_WWDG_DelInit.....	603
45.2.6 HAL_WWDG_MspInit.....	603
45.2.7 HAL_WWDG_MspDelInit	603
45.2.8 HAL_WWDG_WakeupCallback	604
45.2.9 HAL_WWDG_Start.....	604
45.2.10 HAL_WWDG_Start_IT.....	604
45.2.11 HAL_WWDG_Refresh.....	605
45.2.12 HAL_WWDG_IRQHandler	605
45.2.13 HAL_WWDG_WakeupCallback	606
45.2.14 HAL_WWDG_GetState	606
45.3 WWDG Firmware driver defines.....	606
45.3.1 WWDG.....	606
46 FAQs.....	609
47 Revision history	613

List of tables

Table 1: Acronyms and definitions	35
Table 2: HAL drivers files	37
Table 3: User-application files	38
Table 4: APis classification	43
Table 5: List of devices supported by HAL drivers	44
Table 6: HAL API naming rules	46
Table 7: Macros handling interrupts and specific clock configurations	47
Table 8: Callback functions	49
Table 9: HAL generic APIs	49
Table 10: HAL extension APIs	50
Table 11: Define statements used for HAL configuration	54
Table 12: Description of GPIO_InitTypeDef structure	56
Table 13: Description of EXTI configuration macros	58
Table 14: MSP functions	63
Table 15: Timeout values	67
Table 16: Redirection of COMP outputs to embedded timers	113
Table 17: COMP Inputs for the STM32L1xx devices	113
Table 18: Frame formats	269
Table 19: OPAMPs inverting/non-inverting inputs for STM32L1 devices	313
Table 20: OPAMP outputs for STM32L1 devices	313
Table 21: Number of wait states (WS) according to CPU clock (HCLK) frequency	362
Table 22: Clock frequency versus product voltage range	362
Table 23: Frame formats	461
Table 24: Frame formats	563
Table 25: Frame formats	585
Table 26: Document revision history	613

List of figures

Figure 1: Example of project template	40
Figure 2: Adding device-specific functions	51
Figure 3: Adding family-specific functions	51
Figure 4: Adding new peripherals	52
Figure 5: Updating existing APIs	52
Figure 6: File inclusion model	53
Figure 7: HAL driver model	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 consist of 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.
- Peripheral multi-instance support 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 drivers 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>stm32l1xx_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: stm32l1xx_hal_adc.h, stm32l1xx_hal_irda.h, ...</i>

File	Description
<i>stm32l1xx_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: stm32l1xx_hal_adc_ex.c, stm32l1xx_hal_dma_ex.c, ...</i>
<i>stm32l1xx_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: stm32l1xx_hal_adc_ex.h, stm32l1xx_hal_dma_ex.h, ...</i>
<i>stm32l1xx_hal.c</i>	This file is used for HAL initialization and contains DBGMCU, Remap and Time Delay based on systick APIs.
<i>stm32l1xx_hal.h</i>	<i>stm32l1xx_hal.c</i> header file
<i>stm32l1xx_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>stm32l1xx_hal_conf_template.h</i>	Template file allowing to customize the drivers for a given application.
<i>stm32l1xx_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_stm32l1xx.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 to : <ul style="list-style-type: none">• relocate the vector table in internal SRAM.
<i>startup_stm32l1xx.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>stm32l1xx_flash.icf (optional)</i>	Linker file for EWARM toolchain allowing mainly to adapt the stack/heap size to fit the application requirements.
<i>stm32l1xx_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>stm32l1xx_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>stm32l1xx_it.c/h</i>	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>stm32l1xx_hal.c</i>) used as HAL timebase. By default, this function is called each 1ms in Systick ISR. . . The PPP_IRQHandler() routine must call HAL_PPP_IRQHandler() if an interrupt based process is used within the application.
<i>main.c/h</i>	This file contains the main program routine, mainly: <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.

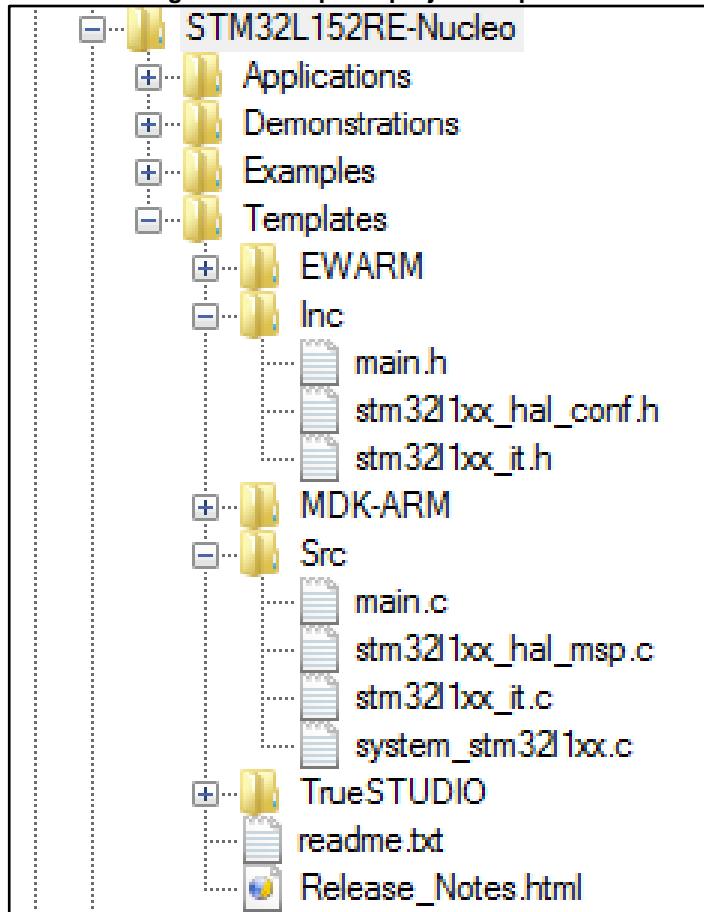
Each project template provides empty main loop function and can be used as a starting point to get familiar with project settings for STM32Cube. Their characteristics are the following:

- It contains 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 to configure 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_InitTypeDef;
```



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 drivers 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 SingleDiff);
uint32_t HAL_ADCEx_Calibration_GetValue(ADC_HandleTypeDef* hadc, uint32_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: APIs 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

IP/module	STM32L100xB	STM32L100xBA	STM32L100xC	STM32L151xB	STM32L151xBA	STM32L151xC	STM32L151xCA	STM32L151xD	STM32L151xE	STM32L152xB	STM32L152xBA	STM32L152xC	STM32L152xDA	STM32L152xE	STM32L162xC	STM32L162xCA	STM32L162xD	STM32L162xE
	VALUE LINE			ACCESS LINE						LCD LINE without AES						LCD Line with AES		
stm32l1xx_hal.c	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_adc.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_adc_ex.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_comp.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_cortex.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_crc.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_cryp.c/h	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	Yes
stm32l1xx_hal_dac.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_dac_ex.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_dma.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_flash.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_flash_ramfunc.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_flash_ex.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_gpio.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_i2c.c/h	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
stm32l1xx_hal_i2s.c/h	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_irda.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_iwdg.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_lcd.c/h	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	Yes

IP/module	STM32L100xB	STM32L100xBA	STM32L100xC	STM32L151xB	STM32L151xBA	STM32L151xC	STM32L151xCA	STM32L151xD	STM32L151xE	STM32L152xB	STM32L152xBA	STM32L152xC	STM32L152xCA	STM32L152xD	STM32L152xE	STM32L162xC	STM32L162xCA	STM32L162xD	STM32L162xE
	VALUE LINE			ACCESS LINE						LCD LINE without AES						LCD Line with AES			
stm32l1xx_hal_msp_template.c	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
stm32l1xx_hal_nor.c/h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No	No	Yes	No
stm32l1xx_hal_opamp.c/h	No	No	No	No	No	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_opamp_ex.c/h	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_pcd.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_pcd_ex.c/h	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
stm32l1xx_hal_pwr_ex.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_rcc.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_rcc_ex.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_rtc.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_rtc_ex.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_sd.c/h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No	No	Yes	No
stm32l1xx_hal_smartcard.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_spi.c/h	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
stm32l1xx_hal_sram.c/h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No	No	Yes	No
stm32l1xx_hal_tim.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_tim_ex.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_uart.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_hal_usart.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

IP/module	STM32L100xB	STM32L100xBA	STM32L100xC	STM32L151xB	STM32L151xBA	STM32L151xC	STM32L151xCA	STM32L151xD	STM32L151xE	STM32L152xB	STM32L152xBA	STM32L152xC	STM32L152xCA	STM32L152xD	STM32L152xE	STM32L162xC	STM32L162xCA	STM32L162xD	STM32L162xE
	VALUE LINE				ACCESS LINE					LCD LINE without AES						LCD Line with AES			
stm32l1xx_hal_wwdg.c/h	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
stm32l1xx_ll_fsmc.c/h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No	No	Yes	No
stm32l1xx_ll_sdmmc.c/h	No	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	No	No	Yes	No

2.5 HAL drivers 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	stm32l1xx_hal_ppp (c/h)	stm32l1xx_hal_ppp_ex (c/h)	stm32l1xx_hal_ppp_ex (c/h)
Module name	HAL_PPP_MODULE		
Function name	HAL_PPP_Function HAL_PPP_FeatureFunction_MODE	HAL_PPPEx_Function HAL_PPPEx_FeatureFunction_MODE	HAL_PPPEx_Function HAL_PPPEx_FeatureFunction_MODE
Handle name	PPP_HandleTypeDef	NA	NA
Init structure name	PPP_InitTypeDef	NA	PPP_InitTypeDef
Enum name	HAL_PPP_StructnameTypeDef	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, stm32l152xb.h, stm32l152xba.h,

- stm32l152xc.h, stm32l152xca.h, stm32l152xd.h, stm32l152xe.h, stm32l162xc.h,
stm32l162xca.h, stm32l162xd.h or stm32l162xe.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_DeInit, e.g. TIM_DeInit.
 - 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

Macros	Description
<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 MsplInit()` and `HAL PPP MspDelInit`
- Process complete callbacks : `HAL PPP ProcessCpltCallback`
- Error callback: `HAL PPP ErrorCallback`.

Table 8: Callback functions

Callback functions	Example
HAL_PPP_MspInit() / _DelInit()	Ex: HAL_USART_MspInit() Called from HAL_PPP_Init() API function to perform peripheral system level initialization (GPIOs, clock, DMA, interrupt)
HAL_PPP_ProcessCpltCallback	Ex: HAL_USART_TxCpltCallback Called by peripheral or DMA interrupt handler when the process completes
HAL_PPP_ErrorCallback	Ex: HAL_USART_ErrorCallback 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_DelInit()
- **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_DelInit()* 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>	HAL_ADC_Init()	This function initializes the peripheral and configures the low-level resources (clocks, GPIO, AF..)
	HAL_ADC_DelInit()	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>	HAL_ADC_Start()	This function starts ADC conversions when the polling method is used

Function Group	Common API Name	Description
	<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
	<i>HAL_ADC_ConvCpltCallback()</i>	Callback function called in the IT subroutine to indicate the end of the current process or when a DMA transfer has completed
	<i>HAL_ADC_ErrorCallback()</i>	Callback function called in the IT subroutine if a peripheral error or a DMA transfer error occurred
<i>Control group</i>	<i>HAL_ADC_ConfigChannel()</i>	This function configures the selected ADC regular channel, the corresponding rank in the sequencer and the sample time
	<i>HAL_ADC_AnalogWDGConfig</i>	This function configures the analog watchdog for the selected ADC
<i>State and Errors group</i>	<i>HAL_ADC_GetState()</i>	This function allows getting in runtime the peripheral and the data flow states.
	<i>HAL_ADC_GetError()</i>	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
<i>HAL_ADCEx_CalibrationStart()</i>	This function is used to start the automatic ADC calibration
<i>HAL_ADCEx_Calibration_GetValue()</i>	This function is used to get the ADC calibration factor
<i>HAL_ADCEx_Calibration_SetValue()</i>	This function is used to set the calibration factor to overwrite automatic conversion result

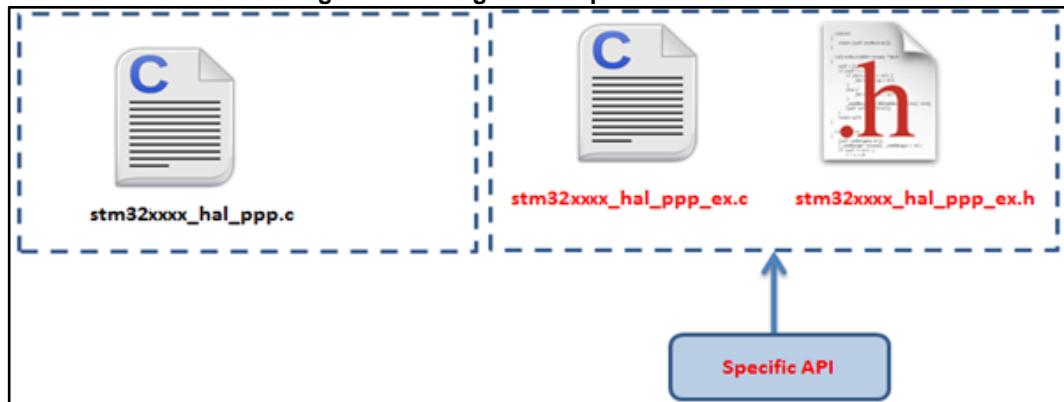
2.7.2 HAL extension model cases

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

Case1: 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_PPPEEx_Function()`.

Figure 2: Adding device-specific functions



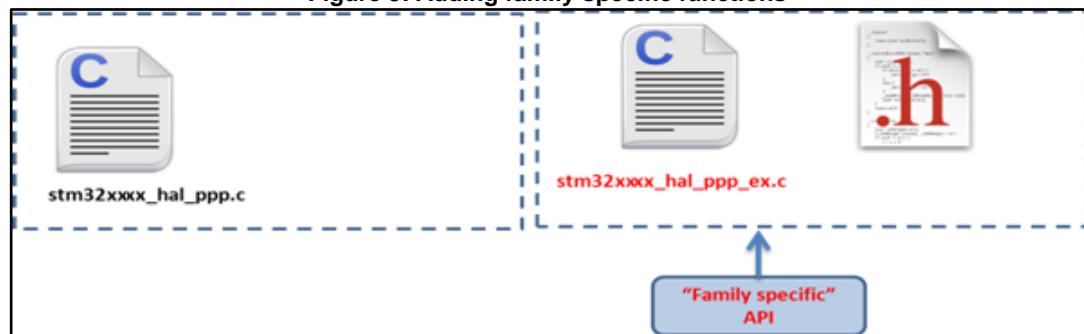
Example: `stm32l1xx_hal_rcc_ex.c/h`

```
#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_RCCEEx_EnableLSECSS(void);
void HAL_RCCEEx_DisableLSECSS(void);
#endif /* STM32L100xBA || .... STM32L162xE*/
```

Case2: Adding a family-specific function

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

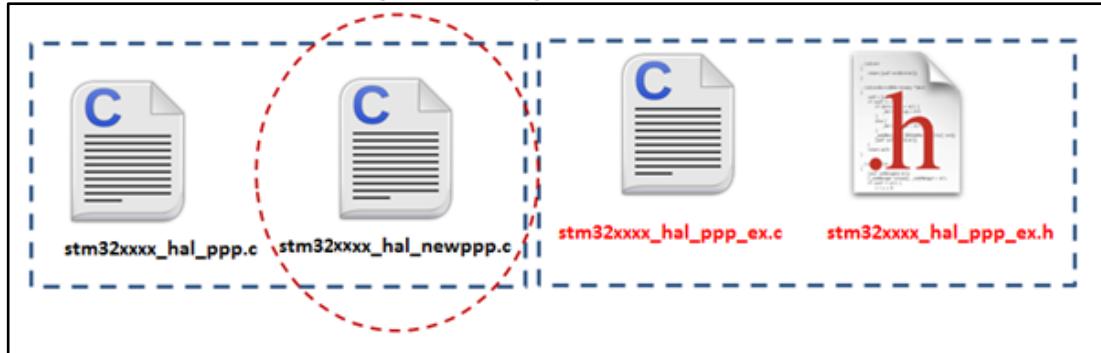
Figure 3: Adding family-specific functions



Case3 : 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_hal_conf.h` using the macro:

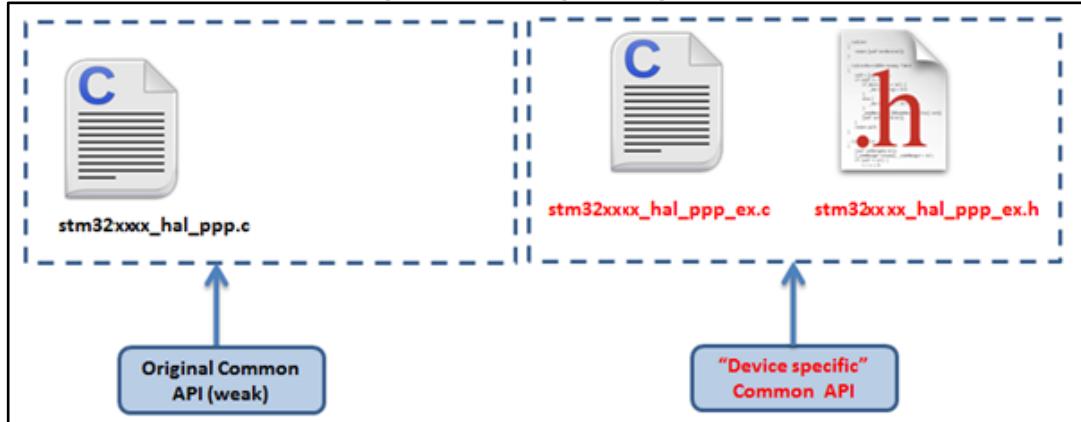
```
#define HAL_NEWPPP_MODULE_ENABLED
```

Figure 4: Adding new peripherals

Example: `stm32l1xx_hal_lcd.c/h`

Case4: 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

Case5 : Updating existing data structures

The data structure for a specific device part number (e.g. `PPP_InitTypeDef`) can have 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:

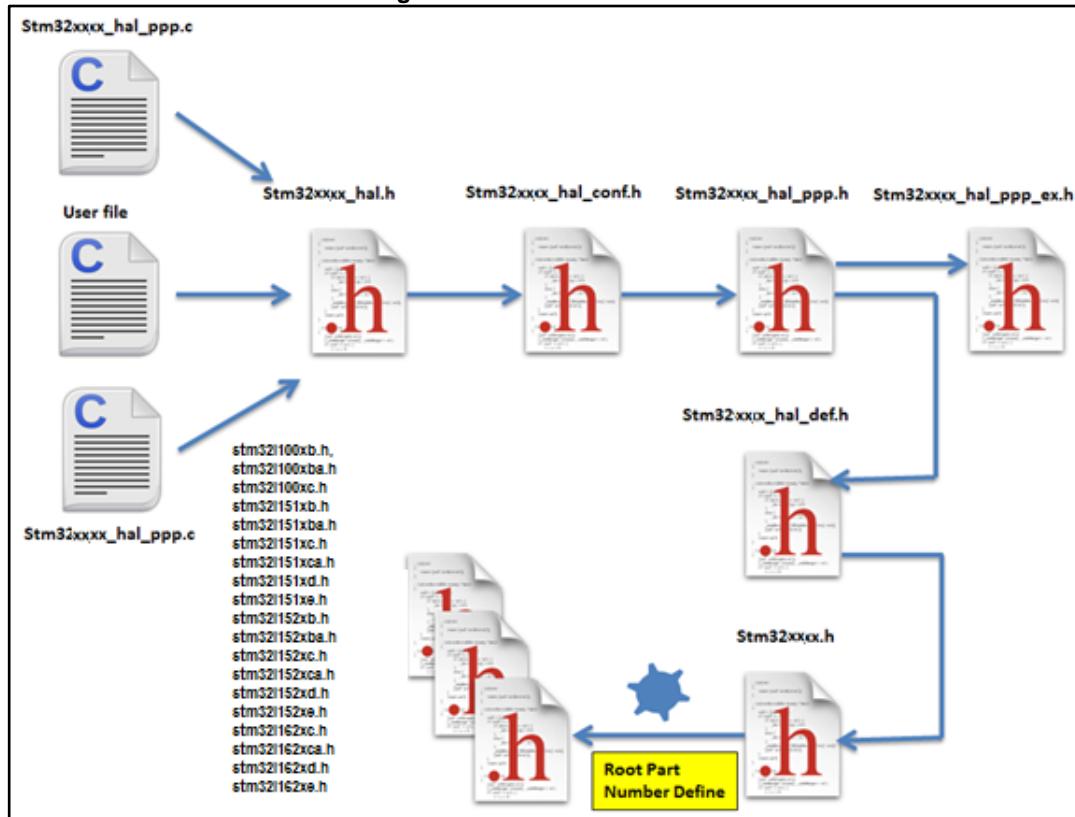
```
#if defined (STM32L100xB)
typedef struct
{
(...)

}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 stm32l1xx_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**
 - Macros defining NULL and HAL_MAX_DELAY

```
#ifndef NULL
#define NULL (void *) 0
#endif
#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)              |
| USE_RTOS            | 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_DelInit()` Clock de-init function that return 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`. They allows 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), 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.<br>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.<br>Possible values are: <ul style="list-style-type: none"> <li>• <u>GPIO mode</u> <ul style="list-style-type: none"> <li>– GPIO_MODE_INPUT : Input Floating</li> <li>– GPIO_MODE_OUTPUT_PP : Output Push Pull</li> <li>– GPIO_MODE_OUTPUT_OD : Output Open Drain</li> <li>– GPIO_MODE_AF_PP : Alternate Function Push Pull</li> <li>– GPIO_MODE_AF_OD : Alternate Function Open Drain</li> <li>– GPIO_MODE_ANALOG : Analog mode</li> </ul> </li> <li>• <u>External Interrupt Mode</u> <ul style="list-style-type: none"> <li>– GPIO_MODE_IT_RISING : Rising edge trigger detection</li> <li>– GPIO_MODE_IT_FALLING : Falling edge trigger detection</li> <li>– GPIO_MODE_IT_RISING_FALLING : Rising/Falling edge trigger detection</li> </ul> </li> <li>• <u>External Event Mode</u> <ul style="list-style-type: none"> <li>– GPIO_MODE_EVT_RISING : Rising edge trigger detection</li> <li>– GPIO_MODE_EVT_FALLING : Falling edge trigger detection</li> <li>– GPIO_MODE_EVT_RISING_FALLING: Rising/Falling edge trigger detection</li> </ul> </li> </ul> |
| Pull            | Specifies the Pull-up or Pull-down activation for the selected pins.<br>Possible values are:<br>GPIO_NOPULL<br>GPIO_PULLUP<br>GPIO_PULLDOWN                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Speed           | Specifies the speed for the selected pins<br>Possible values are:<br>GPIO_SPEED_VERY_LOW<br>GPIO_SPEED_LOW<br>GPIO_SPEED_MEDIUM<br>GPIO_SPEED_HIGH                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

| Structure field | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alternate       | <p>Peripheral to be connected to the selected pins.<br/>Possible values: GPIO_AFx_PPP, where<br/>AFx: is the alternate function index<br/>PPP: is the peripheral instance<br/>Example: use GPIO_AF1_TIM2 to connect TIM2 IOs on AF1.<br/>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)

- Ultra low 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.<br>Example:<br><code>#define PWR_EXTI_LINE_PVD ((uint32_t)0x00010000)<br/>/*!&lt;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<br>Example:<br><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.<br>Example:<br><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.<br>Example:<br><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.<br>Example:<br><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.<br>Example:<br><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”).



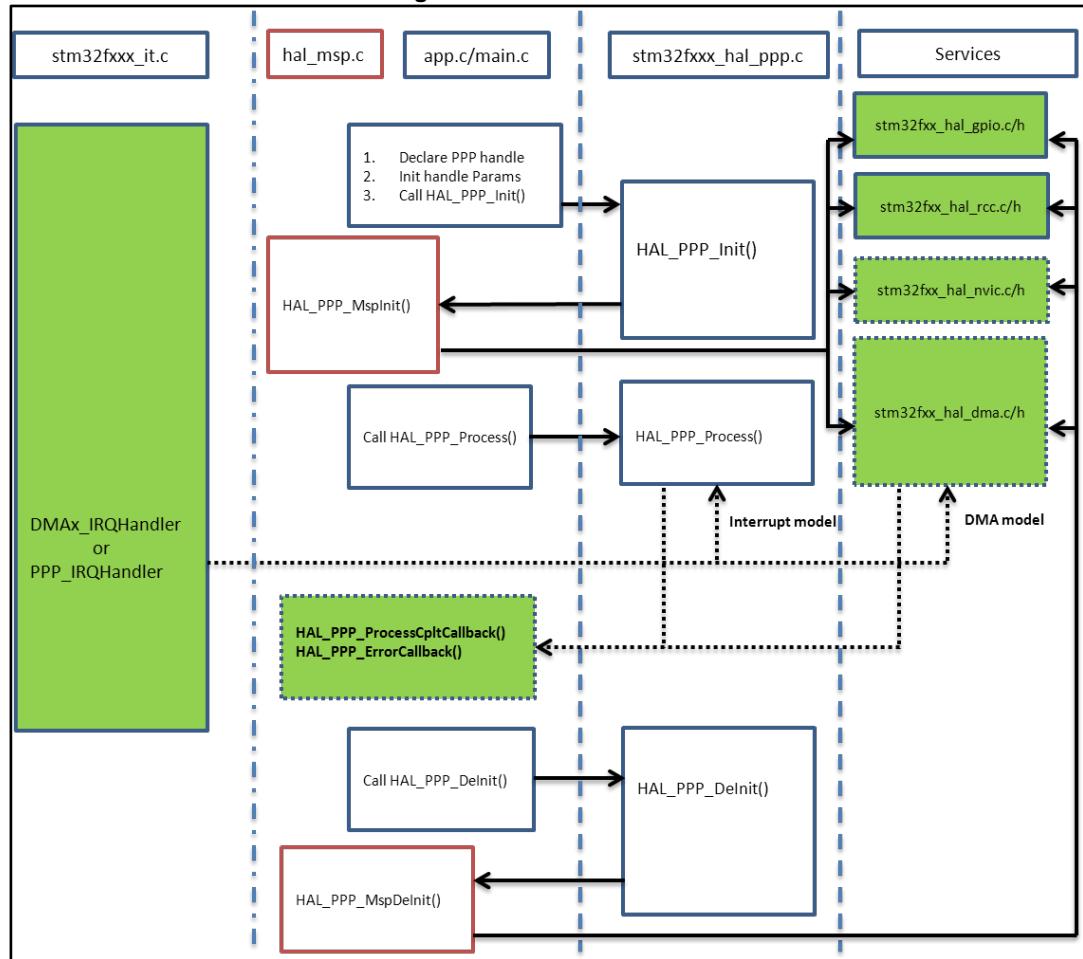
DMA channel callbacks need to be initialized by the user application only in case of memory-to-memory transfer. However when peripheral-to-memory transfers are used, these callbacks are automatically initialized by calling a process API function that uses the DMA.

## 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



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\_DelInit()
  - Resets all peripherals
  - Calls function HAL\_MspDelInit() 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_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.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_HandleTypeDefDef *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_HandleTypeDefDef *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                          |
|---------------------------------|--------------------------------------|
| <b>void HAL_MspInit()</b>       | Global MSP initialization routine    |
| <b>void HAL_MspDeInit()</b>     | Global MSP de-initialization routine |
| <b>void HAL_PPP_MspInit()</b>   | PPP MSP initialization routine       |
| <b>void HAL_PPP_MspDeInit()</b> | 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_tSize,uint32_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 *stm32f1xx\_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.cfile:*

```

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 *stm32l1xx\_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_TypeDef *Instance; /* Register base address */
 PPP_InitTypeDef Init; /* PPP communication parameters */
 HAL_StateTypeDef 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 = UART1;
 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_Handle_tx, hdma_tx);
 __HAL_LINKDMA(UartHandle, DMA_Handle_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.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)
{
}
```

*stm321xx\_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 ... (HAL_MAX_DELAY -1) <sup>(1)</sup> | Timeout in ms                              |
| HAL_MAX_DELAY                           | Infinite poll till process is successful   |

**Notes:**

<sup>(1)</sup>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 for the following items:

- Valid parameters: for some process the used parameters should be valid and already defined, otherwise the system can crash or go into an undefined state. These critical parameters are checked before they are 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 HAL_PPP_StateTypeDef State; /* PPP state */
__IO HAL_PPP_ErrorTypeDef 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 (happ); /* 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 drivers functions. The run-time checking is achieved by using an `assert_param` macro. This macro is used in all the HAL drivers' 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:

```
#ifdef 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 HAL System Driver

### 3.1 HAL Firmware driver API description

The following section lists the various functions of the HAL library.

#### 3.1.1 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

#### 3.1.2 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.
- [\*\*HAL\\_Init\(\)\*\*](#)
- [\*\*HAL\\_DeInit\(\)\*\*](#)
- [\*\*HAL\\_MspInit\(\)\*\*](#)
- [\*\*HAL\\_MspDeInit\(\)\*\*](#)
- [\*\*HAL\\_InitTick\(\)\*\*](#)

#### 3.1.3 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
- *HAL\_IncTick()*
- *HAL\_GetTick()*
- *HAL\_Delay()*
- *HAL\_SuspendTick()*
- *HAL\_ResumeTick()*
- *HAL\_GetHalVersion()*
- *HAL\_GetREVID()*
- *HAL\_GetDEVID()*
- *HAL\_EnableDBGSleepMode()*
- *HAL\_DisableDBGSleepMode()*
- *HAL\_EnableDBGStopMode()*
- *HAL\_DisableDBGStopMode()*
- *HAL\_EnableDBGStandbyMode()*
- *HAL\_DisableDBGStandbyMode()*

### 3.1.4 HAL\_Init

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_Init ( void )</b>                                                                                                                                                                                                                                                                                                                                                                             |
| 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"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• This function is called at the beginning of program after reset and before the clock configuration</li> <li>• 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.</li> </ul> |

### 3.1.5 HAL\_DeInit

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_DeInit ( void )</b>                                           |
| 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"><li>• <b>HAL status</b></li></ul>          |
| Notes         | <ul style="list-style-type: none"><li>• This function is optional.</li></ul> |

### 3.1.6 HAL\_MspInit

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_MspInit ( void )</b>                        |
| Function Description | Initializes the MSP.                                    |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 3.1.7 HAL\_MspDeInit

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_MspDeInit ( void )</b>                      |
| Function Description | DeInitializes the MSP.                                  |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 3.1.8 HAL\_InitTick

|                      |                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_InitTick ( uint32_t TickPriority)</b>                                                                                                                                                                                                                                                                                         |
| Function Description | This function configures the source of the time base.                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"><li>• <b>TickPriority</b> : Tick interrupt priority.</li></ul>                                                                                                                                                                                                                                                       |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"><li>• 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().</li><li>• In the default implementation, SysTick timer is the source of time base. It is used to generate interrupts at regular time</li></ul> |

intervals. Care must be taken if HAL\_Delay() is called from a peripheral ISR process. 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.

### 3.1.9 HAL\_IncTick

|                      |                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_IncTick ( void )</b>                                                                                                                                                                                                                  |
| Function Description | This function is called to increment a global variable "uwTick" used as application time base.                                                                                                                                                    |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>In the default implementation, this variable is incremented each 1ms in Systick ISR.</li><li>This function is declared as __weak to be overwritten in case of other implementations in user file.</li></ul> |

### 3.1.10 HAL\_GetTick

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

### 3.1.11 HAL\_Delay

|               |                                              |
|---------------|----------------------------------------------|
| Function Name | <b>void HAL_Delay ( __IO uint32_t Delay)</b> |
|---------------|----------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | This function provides accurate delay (in milliseconds) based on variable incremented.                                                                                                                                                                                                                                                              |
| 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>• None.</li> </ul>                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> <li>• This function is declared as <code>__weak</code> to be overwritten in case of other implementations in user file.</li> </ul> |

### 3.1.12 HAL\_SuspendTick

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

### 3.1.13 HAL\_ResumeTick

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

### 3.1.14 HAL\_GetHalVersion

|                      |                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_GetHalVersion ( void )</code>                                                    |
| Function Description | Returns the HAL revision.                                                                           |
| Return values        | <ul style="list-style-type: none"><li>version : 0xXYZR (8bits for each decimal, R for RC)</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                                               |

### 3.1.15 HAL\_GetREVID

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_GetREVID ( void )</code>                                |
| Function Description | Returns the device revision identifier.                                    |
| Return values        | <ul style="list-style-type: none"><li>Device revision identifier</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                      |

### 3.1.16 HAL\_GetDEVID

|                      |                                                                   |
|----------------------|-------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_GetDEVID ( void )</code>                       |
| Function Description | Returns the device identifier.                                    |
| Return values        | <ul style="list-style-type: none"><li>Device identifier</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>             |

### 3.1.17 HAL\_EnableDBGSleepMode

|               |                                                   |
|---------------|---------------------------------------------------|
| Function Name | <code>void HAL_EnableDBGSleepMode ( void )</code> |
|---------------|---------------------------------------------------|

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Description | Enable the Debug Module during SLEEP mode.              |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 3.1.18 HAL\_DisableDBGSleepMode

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_DisableDBGSleepMode ( void )</b>            |
| Function Description | Disable the Debug Module during SLEEP mode.             |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 3.1.19 HAL\_EnableDBGStopMode

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_EnableDBGStopMode ( void )</b>              |
| Function Description | Enable the Debug Module during STOP mode.               |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 3.1.20 HAL\_DisableDBGStopMode

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_DisableDBGStopMode ( void )</b>             |
| Function Description | Disable the Debug Module during STOP mode.              |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 3.1.21 HAL\_EnableDBGStandbyMode

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_EnableDBGStandbyMode ( void )</b>           |
| Function Description | Enable the Debug Module during STANDBY mode.            |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 3.1.22 HAL\_DisableDBGStandbyMode

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_DisableDBGStandbyMode ( void )</b>          |
| Function Description | Disable the Debug Module during STANDBY mode.           |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

## 3.2 HAL Firmware driver defines

### 3.2.1 HAL

HAL

*HAL Private Defines*

- **\_STM32L1xx\_HAL\_VERSION\_MAIN**  
[31:24] main version
- **\_STM32L1xx\_HAL\_VERSION\_SUB1**  
[23:16] sub1 version
- **\_STM32L1xx\_HAL\_VERSION\_SUB2**  
[15:8] sub2 version
- **\_STM32L1xx\_HAL\_VERSION\_RC**  
[7:0] release candidate
- **\_STM32L1xx\_HAL\_VERSION**
- **IDCODE\_DEVID\_MASK**

## 4 HAL ADC Generic Driver

### 4.1 ADC Firmware driver registers structures

#### 4.1.1 ADC\_InitTypeDef

*ADC\_InitTypeDef* is defined in the `stm32l1xx_hal_adc.h`

##### 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 low 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 intending to be used in ADC mode 'interruption' (function **HAL\_ADC\_Start\_IT()**), parameter EOCSelection must be set to each conversion (this is not needed for ADC mode 'transfer by DMA', with function **HAL\_ADC\_Start\_DMA()**)
- **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. 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. 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. 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. This parameter can be set to ENABLE or DISABLE. 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 lauch a conversion).

#### 4.1.2 ADC\_ChannelConfTypeDef

**`ADC_ChannelConfTypeDef`** is defined in the `stm32l1xx_hal_adc.h`

##### 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).

#### 4.1.3 ADC\_AnalogWDGConfTypeDef

**ADC\_AnalogWDGConfTypeDef** is defined in the `stm32l1xx_hal_adc.h`

##### 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

#### 4.1.4 ADC\_HandleTypeDef

*ADC\_HandleTypeDef* is defined in the `stm32l1xx_hal_adc.h`

##### Data Fields

- *ADC\_TypeDef \* Instance*
- *ADC\_InitTypeDef Init*
- *\_\_IO uint32\_t NbrOfConversionRank*
- *DMA\_HandleTypeDef \* DMA\_Handle*
- *HAL\_LockTypeDef Lock*
- *\_\_IO HAL\_ADC\_StateTypeDef State*
- *\_\_IO uint32\_t ErrorCode*

##### Field Documentation

- *ADC\_TypeDef\* 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 HAL\_ADC\_StateTypeDef ADC\_HandleTypeDef::State* ADC communication state
- *\_\_IO uint32\_t ADC\_HandleTypeDef::ErrorCode* ADC Error code

## 4.2 ADC Firmware driver API description

The following section lists the various functions of the ADC library.

### 4.2.1 ADC specific 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 automatic conversion of channel 0 to channel 'n'.
- Data alignment with in-built data coherency.
- Channel-wise programmable sampling time.
- ADC conversion Regular or Injected groups.
- 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 calibration
- 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).

## 4.2.2 How to use this driver

1. Enable the ADC interface As prerequisite, ADC clock must be configured at RCC top level. Two clocks settings are mandatory:
  - ADC clock (core clock): Example: Into HAL\_ADC\_MspInit() (recommended code location): \_\_ADC1\_CLK\_ENABLE();
  - ADC clock (conversions clock): Only one possible clock source: derived from HSI RC 16MHz oscillator (HSI).
    - Example: Into HAL\_ADC\_MspInit() or with main setting of RCC:
 

```
RCC_OscInitTypeDef RCC_OscInitStruct;
HAL_RCC_GetOscConfig(&RCC_OscInitStruct);
RCC_OscInitStruct.OscillatorType = (... | RCC OSCILLATORTYPE_HSI); RCC_OscInitStruct.HSIStrate = RCC_HSI_ON; RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT; RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE; RCC_OscInitStruct.PLL.PLLSource = ...
RCC_OscInitStruct.PLL.PLL...
HAL_RCC_OscConfig(&RCC_OscInitStruct);
```
    - 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 ".HSIStrate = RCC\_HSI\_ON"
    - 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 the following function: \_\_GPIOx\_CLK\_ENABLE();
  - Configure these ADC pins in analog mode using HAL\_GPIO\_Init();
3. Configure the ADC parameters (conversion resolution, data alignment, continuous mode, ...) using the HAL\_ADC\_Init() function.
4. Activate the ADC peripheral using one of the start functions: HAL\_ADC\_Start(), HAL\_ADC\_Start\_IT(), HAL\_ADC\_Start\_DMA().

### Channels configuration to regular group

- To configure the ADC regular group features, use HAL\_ADC\_Init() and HAL\_ADC\_ConfigChannel() functions.
- To read the ADC converted values, use the HAL\_ADC\_GetValue() function.

### DMA for regular group configuration

- To enable the DMA mode for regular group, use the HAL\_ADC\_Start\_DMA() function.
- To enable the generation of DMA requests continuously at the end of the last DMA transfer, use the HAL\_ADC\_Init() function. \*

## 4.2.3 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize and configure the ADC.
- De-initialize the ADC
- [\*HAL\\_ADC\\_Init\(\)\*](#)
- [\*HAL\\_ADC\\_DelInit\(\)\*](#)
- [\*HAL\\_ADC\\_MspInit\(\)\*](#)
- [\*HAL\\_ADC\\_MspDelInit\(\)\*](#)

#### 4.2.4 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.
- [\*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\(\)\*](#)

#### 4.2.5 Peripheral Control functions

This section provides functions allowing to:

- Configure channels on regular group
- Configure the analog watchdog
- [\*HAL\\_ADC\\_ConfigChannel\(\)\*](#)
- [\*HAL\\_ADC\\_AnalogWDGConfig\(\)\*](#)

#### 4.2.6 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

- ***HAL\_ADC\_GetState()***
- ***HAL\_ADC\_GetError()***

#### 4.2.7 HAL\_ADC\_Init

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_Init ( <i>ADC_HandleTypeDef</i> *<br/>hadc)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Function Description | Initializes the ADC peripheral and regular group according to parameters specified in structure "ADC_InitTypeDef".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• 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().</li> <li>• 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".</li> <li>• 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".</li> </ul> |

#### 4.2.8 HAL\_ADC\_DeInit

|                      |                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_DeInit ( <i>ADC_HandleTypeDef</i> *<br/>hadc)</b>                                                                                                                                                           |
| Function Description | Deinitialize the ADC peripheral registers to its default reset values.                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> </ul>                                                                                                                                                             |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• 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().</li> </ul> |

#### 4.2.9 HAL\_ADC\_MspInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_ADC_MspInit ( <i>ADC_HandleTypeDef</i> * hadc)</b>             |
| Function Description | Initializes the ADC MSP.                                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 4.2.10 HAL\_ADC\_MspDelInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_ADC_MspDelInit ( <i>ADC_HandleTypeDef</i> * hadc)</b>          |
| Function Description | Deinitializes the ADC MSP.                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 4.2.11 HAL\_ADC\_Start

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_Start ( <i>ADC_HandleTypeDef</i> * hadc)</b>  |
| Function Description | Enables ADC, starts conversion of regular group.                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 4.2.12 HAL\_ADC\_Stop

|                      |                                                                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_Stop ( <i>ADC_HandleTypeDef</i> * <b>hadc</b>)</b>                                                                                                                                                                           |
| 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"> <li>• <b>hadc</b> : ADC handle</li> </ul>                                                                                                                                                                              |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b>.</li> </ul>                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• : ADC peripheral disable is forcing interruption of potential conversion on injected group. If injected group is under use, it should be preliminarily stopped using HAL_ADCEx_InjectedStop function.</li> </ul> |

#### 4.2.13 HAL\_ADC\_PollForConversion

|                      |                                                                                                                                        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_PollForConversion ( <i>ADC_HandleTypeDef</i> * <b>hadc</b>, uint32_t <b>Timeout</b>)</b>                  |
| Function Description | Wait for regular group conversion to be completed.                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> <li>• <b>Timeout</b> : Timeout value in millisecond.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                  |

Notes

• None.

#### 4.2.14 HAL\_ADC\_PollForEvent

|                      |                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_PollForEvent ( <i>ADC_HandleTypeDef</i> * <b>hadc</b>, uint32_t <b>EventType</b>, uint32_t <b>Timeout</b>)</b>              |
| Function Description | Poll for conversion event.                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> <li>• <b>EventType</b> : the ADC event type. This parameter can be one</li> </ul> |

|               |                                                  |
|---------------|--------------------------------------------------|
|               | of the following values:                         |
|               | – <b>AWD_EVENT</b> : ADC Analog watchdog event.  |
|               | – <b>OVR_EVENT</b> : ADC Overrun event           |
|               | • <b>Timeout</b> : Timeout value in millisecond. |
| Return values | • <b>HAL status</b>                              |
| Notes         | • None.                                          |

#### 4.2.15 HAL\_ADC\_Start\_IT

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_Start_IT (</b><br><b><i>ADC_HandleTypeDef* hadc</i></b> ) |
| Function Description | Enables ADC, starts conversion of regular group with interruption.                     |
| Parameters           | • <b>hadc</b> : ADC handle                                                             |
| Return values        | • <b>HAL status</b>                                                                    |
| Notes                | • None.                                                                                |

#### 4.2.16 HAL\_ADC\_Stop\_IT

|                      |                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_Stop_IT (</b><br><b><i>ADC_HandleTypeDef* hadc</i></b> )                                                                        |
| 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           | • <b>hadc</b> : ADC handle                                                                                                                                   |
| Return values        | • None.                                                                                                                                                      |
| Notes                | • None.                                                                                                                                                      |

#### 4.2.17 HAL\_ADC\_Start\_DMA

|                      |                                                                                                                                                                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_Start_DMA (</b><br><b>ADC_HandleTypeDef * hadc, uint32_t * pData, uint32_t Length)</b>                                                                                                             |
| Function Description | Enables ADC, starts conversion of regular group and transfers result through DMA.                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li><b>hadc</b> : ADC handle</li> <li><b>pData</b> : The destination Buffer address.</li> <li><b>Length</b> : The length of data to be transferred from ADC peripheral to memory.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                         |

#### 4.2.18 HAL\_ADC\_Stop\_DMA

|                      |                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_Stop_DMA (</b><br><b>ADC_HandleTypeDef * hadc)</b>                                                                                                                                                                         |
| 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           | <ul style="list-style-type: none"> <li><b>hadc</b> : ADC handle</li> </ul>                                                                                                                                                                              |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status.</b></li> </ul>                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>: ADC peripheral disable is forcing interruption of potential conversion on injected group. If injected group is under use, it should be preliminarily stopped using HAL_ADCEx_InjectedStop function.</li> </ul> |

#### 4.2.19 HAL\_ADC\_GetValue

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_ADC_GetValue (</b><br><b>ADC_HandleTypeDef * hadc)</b>     |
| Function Description | Get ADC regular group conversion result.                                   |
| Parameters           | <ul style="list-style-type: none"> <li><b>hadc</b> : ADC handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>Converted value</b></li> </ul>   |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                    |

#### 4.2.20 HAL\_ADC\_IRQHandler

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_ADC_IRQHandler ( <i>ADC_HandleTypeDef</i> * hadc)</b>          |
| Function Description | Handles ADC interrupt request.                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 4.2.21 HAL\_ADC\_ConvCpltCallback

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_ADC_ConvCpltCallback ( <i>ADC_HandleTypeDef</i> * hadc)</b>    |
| Function Description | Conversion complete callback in non blocking mode.                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 4.2.22 HAL\_ADC\_ConvHalfCpltCallback

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_ADC_ConvHalfCpltCallback ( <i>ADC_HandleTypeDef</i> * hadc)</b> |
| Function Description | Conversion DMA half-transfer callback in non blocking mode.                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li></ul>  |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                     |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                     |

#### 4.2.23 HAL\_ADC\_LevelOutOfWindowCallback

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_ADC_LevelOutOfWindowCallback ( <i>ADC_HandleTypeDef</i> * hadc)</b> |
| Function Description | Analog watchdog callback in non blocking mode.                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> </ul>    |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                       |

#### 4.2.24 HAL\_ADC\_ErrorCallback

|                      |                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_ADC_ErrorCallback ( <i>ADC_HandleTypeDef</i> * hadc)</b>                          |
| Function Description | ADC error callback in non blocking mode (ADC conversion with interruption or transfer by DMA) |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> </ul>                  |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                     |

#### 4.2.25 HAL\_ADC\_ConfigChannel

|                      |                                                                                                                                                                |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_ConfigChannel ( <i>ADC_HandleTypeDef</i> * hadc, <i>ADC_ChannelConfTypeDef</i> * sConfig)</b>                                     |
| Function Description | Configures the the selected channel to be linked to the regular group.                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> <li>• <b>sConfig</b> : Structure of ADC channel for regular group.</li> </ul>           |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• In case of usage of internal measurement channels: Vbat/VrefInt/TempSensor. These internal paths can be be</li> </ul> |

- disabled using function HAL\_ADC\_DeInit().
- 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".

#### 4.2.26 HAL\_ADC\_AnalogWDGConfig

|                      |                                                                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADC_AnalogWDGConfig (</b><br><b>ADC_HandleTypeDef * hadc, ADC_AnalogWDGConfTypeDef *</b><br><b>AnalogWDGConfig)</b>                |
| Function Description | Configures the analog watchdog.                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li><b>hadc</b> : ADC handle</li> <li><b>AnalogWDGConfig</b> : Structure of ADC analog watchdog configuration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                     |

#### 4.2.27 HAL\_ADC\_GetState

|                      |                                                                                    |
|----------------------|------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_ADC_StateTypeDef HAL_ADC_GetState (</b><br><b>ADC_HandleTypeDef * hadc)</b> |
| Function Description | return the ADC state                                                               |
| Parameters           | <ul style="list-style-type: none"> <li><b>hadc</b> : ADC handle</li> </ul>         |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL state</b></li> </ul>                 |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                            |

#### 4.2.28 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"><li>• <code>hadc</code> : ADC handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>ADC Error Code</b></li></ul>          |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                          |

### 4.3 ADC Firmware driver defines

#### 4.3.1 ADC

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`
- `IS_ADC_ANALOG_WATCHDOG_MODE`

*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**
- **IS\_ADC\_CHANNEL**

#### ***ADC ChannelsBank***

- **ADC\_CHANNELS\_BANK\_A**
- **ADC\_CHANNELS\_BANK\_B**
- **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
- **IS\_ADC\_CLOCKPRESCALER**

#### ***ADC conversion group***

- **REGULAR\_GROUP**
- **INJECTED\_GROUP**
- **REGULAR\_INJECTED\_GROUP**
- **IS\_ADC\_CONVERSION\_GROUP**

#### ***ADC Data\_align***

- **ADC\_DATAALIGN\_RIGHT**
- **ADC\_DATAALIGN\_LEFT**
- **IS\_ADC\_DATA\_ALIGN**

#### ***ADC EOCSelection***

- **EOC\_SEQ\_CONV**
- **EOC\_SINGLE\_CONV**
- **IS\_ADC\_EOC\_SELECTION**

#### ***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**

- **AWD\_EVENT**  
ADC Analog watchdog event
- **OVR\_EVENT**  
ADC overrun event
- **IS\_ADC\_EVENT\_TYPE**

**ADC Exported Macros**

- **\_HAL\_ADC\_IS\_ENABLED**  
**Description:** Verification of ADC state: enabled or disabled.  
**Parameters:** `_HANDLE_`: ADC handle  
**Return value:**SET: (ADC enabled) or RESET (ADC disabled)
- **\_HAL\_ADC\_IS\_SOFTWARE\_START\_REGULAR**  
**Description:** Test if conversion trigger of regular group is software start or external trigger.  
**Parameters:** `_HANDLE_`: ADC handle  
**Return value:**SET: (software start) or RESET (external trigger)
- **\_HAL\_ADC\_IS\_SOFTWARE\_START\_INJECTED**  
**Description:** Test if conversion trigger of injected group is software start or external trigger.  
**Parameters:** `_HANDLE_`: ADC handle  
**Return value:**SET: (software start) or RESET (external trigger)
- **\_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  
**Return value:**State: of interruption (SET or RESET)
- **\_HAL\_ADC\_ENABLE\_IT**  
**Description:** Enable the ADC end of conversion interrupt.  
**Parameters:** `_HANDLE_`: ADC handle `_INTERRUPT_`: ADC Interrupt  
**Return value:**None:
- **\_HAL\_ADC\_DISABLE\_IT**  
**Description:** Disable the ADC end of conversion interrupt.  
**Parameters:** `_HANDLE_`: ADC handle `_INTERRUPT_`: ADC Interrupt  
**Return value:**None:
- **\_HAL\_ADC\_GET\_FLAG**  
**Description:** Get the selected ADC's flag status.  
**Parameters:** `_HANDLE_`: ADC handle `_FLAG_`: ADC flag  
**Return value:**None:
- **\_HAL\_ADC\_CLEAR\_FLAG**  
**Description:** Clear the ADC's pending flags.  
**Parameters:** `_HANDLE_`: ADC handle `_FLAG_`: ADC flag  
**Return value:**None:
- **\_HAL\_ADC\_CLEAR\_ERRORCODE**  
**Description:** Clear ADC error code (set it to error code: "no error")

**Parameters:** \_\_HANDLE\_\_: ADC handle

**Return value:** None

- **\_\_HAL\_ADC\_RESET\_HANDLE\_STATE**

**Description:** Reset ADC handle state.

**Parameters:** \_\_HANDLE\_\_: ADC handle

**Return value:** None

#### ***ADC External trigger edge Regular***

- **ADC\_EXTERNALTRIGCONVEDGE\_NONE**
- **ADC\_EXTERNALTRIGCONVEDGE\_RISING**
- **ADC\_EXTERNALTRIGCONVEDGE\_FALLING**
- **ADC\_EXTERNALTRIGCONVEDGE\_RISINGFALLING**
- **IS\_ADC\_EXTRIG\_EDGE**

#### ***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**
- **IS\_ADC\_EXTRIG**

#### ***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 Regular group ready status flag
- **ADC\_FLAG\_POSTCONV\_ALL**

#### ***ADC Internal HAL driver Ext trig src Regular***

- **ADC\_EXTERNALTRIG\_T9\_CC2**
- **ADC\_EXTERNALTRIG\_T9\_TRGO**

- **ADC\_EXTERNALTRIG\_T2\_CC3**
- **ADC\_EXTERNALTRIG\_T2\_CC2**
- **ADC\_EXTERNALTRIG\_T3\_TRGO**
- **ADC\_EXTERNALTRIG\_T4\_CC4**
- **ADC\_EXTERNALTRIG\_T2\_TRGO**
- **ADC\_EXTERNALTRIG\_T3\_CC1**
- **ADC\_EXTERNALTRIG\_T3\_CC3**
- **ADC\_EXTERNALTRIG\_T4\_TRGO**
- **ADC\_EXTERNALTRIG\_T6\_TRGO**
- **ADC\_EXTERNALTRIG\_EXT\_IT11**

#### ***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
- **IS\_ADC\_AUTOPOWEROFF**

#### ***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
- **IS\_ADC\_AUTOWAIT**

#### ***ADC Private Constants***

- **ADC\_ENABLE\_TIMEOUT**
- **ADC\_DISABLE\_TIMEOUT**
- **ADC\_TEMPSENSOR\_DELAY\_CPU\_CYCLES**
- **ADC\_STAB\_DELAY\_CPU\_CYCLES**

*ADC range verification*

- **IS\_ADC\_RANGE**

*ADC regular discontinuous mode number verification*

- **IS\_ADC\_REGULAR\_DISCONT\_NUMBER**

*ADC regular nb conv verification*

- **IS\_ADC\_REGULAR\_NB\_CONV**

*ADC regular rank*

- **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**
- **IS\_ADC\_REGULAR\_RANK**

*ADC Resolution*

- **ADC\_RESOLUTION12b**  
ADC 12-bit resolution
- **ADC\_RESOLUTION10b**  
ADC 10-bit resolution
- **ADC\_RESOLUTION8b**  
ADC 8-bit resolution

- **ADC\_RESOLUTION6b**  
ADC 6-bit resolution
- **IS\_ADC\_RESOLUTION**
- **IS\_ADC\_RESOLUTION\_8\_6\_BITS**

#### *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
- **IS\_ADC\_SAMPLE\_TIME**

#### *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**
- **IS\_ADC\_SCAN\_MODE**

## 5 HAL ADC Extension Driver

### 5.1 ADCEx Firmware driver registers structures

#### 5.1.1 ADC\_InjectionConfTypeDef

*ADC\_InjectionConfTypeDef* is defined in the `stm32l1xx_hal_adc_ex.h`

##### 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.

## 5.2 ADCEx Firmware driver API description

The following section lists the various functions of the ADCEx library.

### 5.2.1 How to use this driver

1. Activate the ADC peripheral using one of the start functions:  
**HAL\_ADCEx\_InjectedStart()**, **HAL\_ADCEx\_InjectedStart\_IT()**.

### Channels configuration to injected group

- To configure the ADC Injected channels group features, use **HAL\_ADCEx\_InjectedConfigChannel()** functions.
- To activate the continuous mode, use the **HAL\_ADC\_Init()** function.
- To read the ADC converted values, use the **HAL\_ADCEx\_InjectedGetValue()** function.

## 5.2.2 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.
- ***HAL\_ADCEx\_InjectedStart()***
- ***HAL\_ADCEx\_InjectedStop()***
- ***HAL\_ADCEx\_InjectedPollForConversion()***
- ***HAL\_ADCEx\_InjectedStart\_IT()***
- ***HAL\_ADCEx\_InjectedStop\_IT()***
- ***HAL\_ADCEx\_InjectedGetValue()***
- ***HAL\_ADCEx\_InjectedConvCpltCallback()***

## 5.2.3 Peripheral Control functions

This section provides functions allowing to:

- Configure channels on injected group
- ***HAL\_ADCEx\_InjectedConfigChannel()***

## 5.2.4 HAL\_ADCEx\_InjectedStart

|                      |                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADCEx_InjectedStart (</b><br><b><i>ADC_HandleTypeDef</i> * hadc)</b> |
| Function Description | Enables ADC, starts conversion of injected group.                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> </ul>                  |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                         |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                     |

### 5.2.5 HAL\_ADCEx\_InjectedStop

|                      |                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADCEx_InjectedStop (</b><br><b><i>ADC_HandleTypeDef * hadc</i></b> )                                                                                                                                                              |
| Function Description | Stop conversion of injected channels.                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li></ul>                                                                                                                                                                                 |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"><li>• If ADC must be disabled with this function and if regular conversion is on going, function HAL_ADC_Stop must be used preliminarily.</li><li>• In case of auto-injection mode, HAL_ADC_Stop must be used.</li></ul> |

### 5.2.6 HAL\_ADCEx\_InjectedPollForConversion

|                      |                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADCEx_InjectedPollForConversion (</b><br><b><i>ADC_HandleTypeDef * hadc, uint32_t Timeout</i></b> )        |
| Function Description | Wait for injected group conversion to be completed.                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li><li>• <b>Timeout</b> : Timeout value in millisecond.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                             |

### 5.2.7 HAL\_ADCEx\_InjectedStart\_IT

|                      |                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADCEx_InjectedStart_IT (</b><br><b><i>ADC_HandleTypeDef * hadc</i></b> ) |
| Function Description | Enables ADC, starts conversion of injected group with interruption.                               |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hadc</b> : ADC handle</li></ul>                        |

---

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL status.</b></li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>              |

## 5.2.8 HAL\_ADCEx\_InjectedStop\_IT

|                      |                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_ADCEx_InjectedStop_IT (</b><br><b><i>ADC_HandleTypeDef * hadc</i></b> )                                                                                        |
| Function Description | Stop conversion of injected channels, disable interruption of end-of-conversion.                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> </ul>                                                                                                            |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• If ADC must be disabled with this function and if regular conversion is on going, function HAL_ADC_Stop must be used preliminarily.</li> </ul> |

## 5.2.9 HAL\_ADCEx\_InjectedGetValue

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_ADCEx_InjectedGetValue (</b><br><b><i>ADC_HandleTypeDef * hadc, uint32_t InjectedRank</i></b> )                                                                                                                                                                                                                                                                                                                                                                                                         |
| Function Description | Get ADC injected group conversion result.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> <li>• <b>InjectedRank</b> : the converted ADC injected rank. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>ADC_INJECTED_RANK_1</b> : Injected Channel1 selected</li> <li>– <b>ADC_INJECTED_RANK_2</b> : Injected Channel2 selected</li> <li>– <b>ADC_INJECTED_RANK_3</b> : Injected Channel3 selected</li> <li>– <b>ADC_INJECTED_RANK_4</b> : Injected Channel4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

### 5.2.10 HAL\_ADCEx\_InjectedConvCpltCallback

|                      |                                                                                          |
|----------------------|------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_ADCEx_InjectedConvCpltCallback (<br/>    ADC_HandleTypeDef * hadc)</code> |
| Function Description | Injected conversion complete callback in non blocking mode.                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hadc</b> : ADC handle</li> </ul>             |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                |

### 5.2.11 HAL\_ADCEx\_InjectedConfigChannel

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_ADCEx_InjectedConfigChannel (</code><br><code>    ADC_HandleTypeDef * hadc, ADC_InjectionConfTypeDef *</code><br><code>    sConfigInjected)</code>                                                                                                                                                                                                                                                    |
| 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"> <li>• <b>hadc</b> : ADC handle</li> <li>• <b>sConfigInjected</b> : Structure of ADC injected group and ADC channel for injected group.</li> </ul>                                                                                                                                                                                                                                              |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> </ul> |

## 5.3 ADCEx Firmware driver defines

### 5.3.1 ADCEx

ADCEx

***ADCEx Exported Macros***

- **`_HAL_ADC_CHANNELS_BANK`**

**Description:** Selection of channels bank.

**Parameters:** `_HANDLE_`: ADC handle `_BANK_`: Bank selection. This parameter can be a value of

**Return value:**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 edge Injected***

- `ADC_EXTERNALTRIGINJECCONV_EDGE_NONE`
- `ADC_EXTERNALTRIGINJECCONV_EDGE_RISING`
- `ADC_EXTERNALTRIGINJECCONV_EDGE_FALLING`
- `ADC_EXTERNALTRIGINJECCONV_EDGE_RISINGFALLING`
- `IS_ADC_EXTTRIGINJEC_EDGE`

***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`
- `IS_ADC_EXTTRIGINJEC`

***ADCEx injected nb conv verification***

- `IS_ADC_INJECTED_NB_CONV`

***ADCEx injected rank***

- `ADC_INJECTED_RANK_1`
- `ADC_INJECTED_RANK_2`
- `ADC_INJECTED_RANK_3`
- `ADC_INJECTED_RANK_4`
- `IS_ADC_INJECTED_RANK`

***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**

#### ***ADCEx Private Constants***

- **ADC\_CONVERSIONCLOCKCYCLES\_SAMPLETIME\_4CYCLE5**
- **ADC\_CONVERSIONCLOCKCYCLES\_SAMPLETIME\_9CYCLES**
- **ADC\_CONVERSIONCLOCKCYCLES\_SAMPLETIME\_16CYCLES**
- **ADC\_CONVERSIONCLOCKCYCLES\_SAMPLETIME\_24CYCLES**
- **ADC\_CONVERSIONCLOCKCYCLES\_SAMPLETIME\_48CYCLES**
- **ADC\_CONVERSIONCLOCKCYCLES\_SAMPLETIME\_96CYCLES**
- **ADC\_CONVERSIONCLOCKCYCLES\_SAMPLETIME\_192CYCLES**
- **ADC\_CONVERSIONCLOCKCYCLES\_SAMPLETIME\_384CYCLES**
- **ADC\_TEMPSENSOR\_DELAY\_CPU\_CYCLES**

#### ***ADCEx Private Macro***

- **\_\_ADC\_SQR1\_L**  
**Description:** Set ADC number of ranks into regular channel sequence length.  
**Parameters:** \_NbrOfConversion\_: Regular channel sequence length  
**Return value:**None
- **\_\_ADC\_SQR1\_SQXX**  
**Description:** Set ADC ranks available in register SQR1.  
**Parameters:** \_NbrOfConversion\_: Regular channel sequence length  
**Return value:**None
- **\_\_ADC\_SMPR0**  
**Description:** Set the ADC's sample time for channel numbers between 30 and 31.  
**Parameters:** \_SAMPLETIME\_: Sample time parameter. \_CHANNELNB\_: Channel number.  
**Return value:**None: None
- **\_\_ADC\_SMPR1**  
**Description:** Set the ADC's sample time for channel numbers between 20 and 29.  
**Parameters:** \_SAMPLETIME\_: Sample time parameter. \_CHANNELNB\_: Channel number.  
**Return value:**None: None
- **ADC\_SMPR1\_CHANNEL\_MAX**  
**Description:** Defines the highest channel available in register SMPR1.  
**Parameters:**None:  
**Return value:**None:
- **\_\_ADC\_SMPR2**  
**Description:** Set the ADC's sample time for channel numbers between 10 and 18.  
**Parameters:** \_SAMPLETIME\_: Sample time parameter. \_CHANNELNB\_: Channel number.  
**Return value:**None
- **\_\_ADC\_SMPR3**  
**Description:** Set the ADC's sample time for channel numbers between 0 and 9.  
**Parameters:** \_SAMPLETIME\_: Sample time parameter. \_CHANNELNB\_: Channel number.  
**Return value:**None
- **\_\_ADC\_SQR5\_RK**  
**Description:** Set the selected regular channel rank for rank between 1 and 6.  
**Parameters:** \_CHANNELNB\_: Channel number. \_RANKNB\_: Rank number.  
**Return value:**None
- **\_\_ADC\_SQR4\_RK**  
**Description:** Set the selected regular channel rank for rank between 7 and 12.

- Parameters:** \_CHANNELNB\_: Channel number. \_RANKNB\_: Rank number.  
**Return value:**None:
- **\_ADC\_SQR3\_RK**  
**Description:** Set the selected regular channel rank for rank between 13 and 18.  
**Parameters:** \_CHANNELNB\_: Channel number. \_RANKNB\_: Rank number.  
**Return value:**None:
  - **\_ADC\_SQR2\_RK**  
**Description:** Set the selected regular channel rank for rank between 19 and 24.  
**Parameters:** \_CHANNELNB\_: Channel number. \_RANKNB\_: Rank number.  
**Return value:**None:
  - **\_ADC\_SQR1\_RK**  
**Description:** Set the selected regular channel rank for rank between 25 and 28.  
**Parameters:** \_CHANNELNB\_: Channel number. \_RANKNB\_: Rank number.  
**Return value:**None:
  - **\_ADC\_JSQR\_JL**  
**Description:** Set the injected sequence length.  
**Parameters:** \_JSQR\_JL\_: Sequence length.  
**Return value:**None:
  - **\_ADC\_JSQR\_RK**  
**Description:** Set the selected injected Channel rank (channels sequence starting from 4-JL)  
**Parameters:** \_CHANNELNB\_: Channel number. \_RANKNB\_: Rank number.  
\_JSQR\_JL\_: Sequence length.  
**Return value:**None:
  - **\_ADC\_CR2\_DMACONTREQ**  
**Description:** Enable the ADC DMA continuous request.  
**Parameters:** \_DMACONTREQ\_MODE\_: DMA continuous request mode.  
**Return value:**None:
  - **\_ADC\_CR2\_CONTINUOUS**  
**Description:** Enable ADC continuous conversion mode.  
**Parameters:** \_CONTINUOUS\_MODE\_: Continuous mode.  
**Return value:**None:
  - **\_ADC\_CR2\_MASK\_ADCINIT**  
**Return value:**None:
  - **\_ADC\_CR1\_DISCONTINUOUS\_NUM**  
**Description:** Configures the number of discontinuous conversions for the regular group channels.  
**Parameters:** \_NBR\_DISCONTINUOUS\_CONV\_: Number of discontinuous conversions.  
**Return value:**None:
  - **\_ADC\_CR1\_SCAN**  
**Description:** Enable ADC scan mode to convert multiple ranks with sequencer.  
**Parameters:** \_SCAN\_MODE\_: Scan conversion mode.  
**Return value:**None:
  - **\_ADC\_CONVCYCLES\_MAX\_RANGE**  
**Description:** Get the maximum ADC conversion cycles on all channels.  
**Parameters:** \_HANDLE\_: ADC handle  
**Return value:**ADC: conversion cycles on all channels
  - **\_ADC\_GET\_CLOCK\_PRESCALER\_DECIMAL**  
**Return value:**None:
  - **\_ADC\_SMPR0\_CLEAR**  
**Description:** Clear register SMPR0.  
**Parameters:** \_HANDLE\_: ADC handle  
**Return value:**None:

- **\_\_ADC\_CR2\_CLEAR**  
**Description:** Clear register CR2.  
**Parameters:** \_\_HANDLE\_\_: ADC handle  
**Return value:**None
- **\_\_ADC\_SMPR0\_CHANNEL\_SET**  
**Description:** Set the sampling time of selected channel on register SMPR0 Register SMPR0 availability depends on device category.  
**Parameters:** \_\_HANDLE\_\_: ADC handle \_\_SAMPLETIME\_\_: Sample time parameter.  
\_\_CHANNEL\_\_: Channel number.  
**Return value:**None
- **\_\_ADC\_ENABLE**  
**Description:** Enable the ADC peripheral.  
**Parameters:** \_\_HANDLE\_\_: ADC handle  
**Return value:**None
- **\_\_ADC\_DISABLE**  
**Description:** Disable the ADC peripheral.  
**Parameters:** \_\_HANDLE\_\_: ADC handle  
**Return value:**None

## 6 HAL COMP Generic Driver

### 6.1 COMP Firmware driver registers structures

#### 6.1.1 COMP\_InitTypeDef

**COMP\_InitTypeDef** is defined in the `stm32l1xx_hal_comp.h`

##### 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.

### 6.1.2 COMP\_HandleTypeDef

**COMP\_HandleTypeDef** is defined in the `stm32l1xx_hal_comp.h`

#### 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

## 6.2 COMP Firmware driver API description

The following section lists the various functions of the COMP library.

### 6.2.1 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. Refer to "table1. COMP Inputs" below. 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. Refer to "table 2. COMP Outputs redirection to embedded timers" below.
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 macro `__HAL_COMP_EXTI_GET_FLAG()`. Possible values are:
    - `COMP_EXTI_LINE_COMP1_EVENT`
    - `COMP_EXTI_LINE_COMP2_EVENT`
6. The comparators also offer the possibility to ouput the voltage reference (`VrefInt`), used on inverting inputs, on I/O pin through a buffer. To use it, refer to macro `"__HAL_VREFINT_OUT_ENABLE()"`.

**Table 16: Redirection of COMP outputs to embedded timers**

| COMP1                    | COMP2                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| No redirection to timers | TIM2 IC4<br>TIM2 OCREF CLR<br>TIM3 IC4<br>TIM3 OCREF CLR<br>TIM4 IC4<br>TIM4 OCREF CLR<br>TIM10 IC1 |

**Table 17: COMP Inputs for the STM32L1xx devices**

|                      |                                                                                                                                                                                                                                                        | COMP1                                                                                       | COMP2                                                                         |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| Inverting inputs     | 1/4 VREFINT<br>1/2 VREFINT<br>3/4 VREFINT<br>VREFINT<br>DAC Ch1 OUT (PA4)<br>DAC Ch2 OUT (PA5)<br>I/O: PB3                                                                                                                                             | -<br>-<br>-<br>OK<br>-<br>-<br>-                                                            | OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK                                        |
| Non-inverting inputs | I/O:<br>• PB4, 5, 6, 7 <sup>(1)</sup><br>• PA0, 1, 2, 3, 4, 5, 6, 7 <sup>(2)</sup><br>• PB0, 1, 12, 13, 14, 15<br>• PC0 1, 2, 3, 4, 5<br>• PE7, 8, 9, 10<br>• PF6, 7, 8, 9, 10<br>• OPAMP1 output<br>• OPAMP2 output<br>• OPAMP3 output <sup>(3)</sup> | -<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK<br>OK | OK<br>-<br>-<br>-<br>-<br>-<br>-<br>-<br>-<br>-<br>-<br>-<br>-<br>-<br>-<br>- |

**Notes:**

(1)PA6/7 are available on devices category Cat.3, Cat.4, Cat.5 only.

(2)PA0/1/2/3 are available on devices category Cat.3, Cat.4, Cat.5 only.

(3)Available on devices category Cat.4 only.

## 6.2.2 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 \_\_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\_DelInit() function

### 6.2.3 Initialization and de-initialization functions

This section provides functions to initialize and de-initialize comparators

- [\*HAL\\_COMP\\_Init\(\)\*](#)
- [\*HAL\\_COMP\\_DelInit\(\)\*](#)
- [\*HAL\\_COMP\\_MspInit\(\)\*](#)
- [\*HAL\\_COMP\\_MspDelInit\(\)\*](#)

### 6.2.4 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.

- [\*HAL\\_COMP\\_Start\(\)\*](#)
- [\*HAL\\_COMP\\_Stop\(\)\*](#)
- [\*HAL\\_COMP\\_Start\\_IT\(\)\*](#)
- [\*HAL\\_COMP\\_Stop\\_IT\(\)\*](#)
- [\*HAL\\_COMP\\_IRQHandler\(\)\*](#)

### 6.2.5 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).

- [\*HAL\\_COMP\\_Lock\(\)\*](#)
- [\*HAL\\_COMP\\_GetOutputLevel\(\)\*](#)
- [\*HAL\\_COMP\\_TriggerCallback\(\)\*](#)

### 6.2.6 Peripheral State functions

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

- [\*HAL\\_COMP\\_GetState\(\)\*](#)

## 6.2.7 HAL\_COMP\_Init

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

## 6.2.8 HAL\_COMP\_DelInit

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

## 6.2.9 HAL\_COMP\_MspInit

|                      |                                                                                |
|----------------------|--------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_COMP_MspInit ( <i>COMP_HandleTypeDef</i> * hcomp)</b>              |
| Function Description | Initializes the COMP MSP.                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hcomp</b> : COMP handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                      |

### 6.2.10 HAL\_COMP\_MspDeInit

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_COMP_MspDeInit ( <i>COMP_HandleTypeDef</i> * hcomp)</b>          |
| Function Description | Deinitializes COMP MSP.                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcomp</b> : COMP handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                      |

### 6.2.11 HAL\_COMP\_Start

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_COMP_Start ( <i>COMP_HandleTypeDef</i> * hcomp)</b> |
| Function Description | Start the comparator.                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcomp</b> : COMP handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>          |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                      |

### 6.2.12 HAL\_COMP\_Stop

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_COMP_Stop ( <i>COMP_HandleTypeDef</i> * hcomp)</b>  |
| Function Description | Stop the comparator.                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcomp</b> : COMP handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>          |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                      |

### 6.2.13 HAL\_COMP\_Start\_IT

|                      |                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_COMP_Start_IT (</b><br><b><i>COMP_HandleTypeDef * hcomp</i></b> ) |
| Function Description | Enables the interrupt and starts the comparator.                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcomp</b> : COMP handle</li></ul>               |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status.</b></li></ul>                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                    |

### 6.2.14 HAL\_COMP\_Stop\_IT

|                      |                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_COMP_Stop_IT (</b><br><b><i>COMP_HandleTypeDef * hcomp</i></b> ) |
| Function Description | Disable the interrupt and Stop the comparator.                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcomp</b> : COMP handle</li></ul>              |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                   |

### 6.2.15 HAL\_COMP\_IRQHandler

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_COMP_IRQHandler (</b><br><b><i>COMP_HandleTypeDef * hcomp</i></b> ) |
| Function Description | Comparator IRQ Handler.                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcomp</b> : COMP handle</li></ul>    |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                         |

### 6.2.16 HAL\_COMP\_Lock

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_COMP_Lock (</b><br><b><i>COMP_HandleTypeDef * hcomp</i></b> ) |
| Function Description | Lock the selected comparator configuration.                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcomp</b> : COMP handle</li></ul>           |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                |

### 6.2.17 HAL\_COMP\_GetOutputLevel

|                      |                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_COMP_GetOutputLevel (</b><br><b><i>COMP_HandleTypeDef * hcomp</i></b> ) |
| Function Description | Return the output level (high or low) of the selected comparator.                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                 |

### 6.2.18 HAL\_COMP\_TriggerCallback

|                      |                                                                                      |
|----------------------|--------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_COMP_TriggerCallback (</b><br><b><i>COMP_HandleTypeDef * hcomp</i></b> ) |
| Function Description | Comparator callback.                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcomp</b> : COMP handle</li></ul>         |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                              |

## 6.2.19 HAL\_COMP\_GetState

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_COMP_StateTypeDef HAL_COMP_GetState (</b><br><b>COMP_HandleTypeDef * hcomp)</b> |
| Function Description | Return the COMP state.                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hcomp</b> : : COMP handle</li> </ul>       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>                   |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                              |

## 6.3 COMP Firmware driver defines

### 6.3.1 COMP

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.:
- **\_HAL\_COMP\_DISABLE**  
**Description:** Disables the specified comparator.  
**Parameters:** `_HANDLE_`: COMP handle.  
**Return value:**None.:
- **\_HAL\_COMP\_GET\_FLAG**  
**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_LOCK`: lock flag  
**Return value:**The: new state of `_FLAG_` (TRUE or FALSE).
- **\_HAL\_COMP\_EXTI\_RISING\_IT\_ENABLE**  
**Description:** Enable the Exti Line rising edge trigger.  
**Parameters:** `_EXTILINE_`: specifies the COMP Exti sources to be enabled. This parameter can be a value of  
**Return value:**None.:
- **\_HAL\_COMP\_EXTI\_RISING\_IT\_DISABLE**  
**Description:** Disable the Exti Line rising edge trigger.  
**Parameters:** `_EXTILINE_`: specifies the COMP Exti sources to be disabled. This parameter can be a value of  
**Return value:**None.:

- **`_HAL_COMP_EXTI_FALLING_IT_ENABLE`**  
**Description:** Enable the Exti Line falling edge trigger.  
**Parameters:** `_EXTILINE_`: specifies the COMP Exti sources to be enabled. This parameter can be a value of  
**Return value:**None.:
- **`_HAL_COMP_EXTI_FALLING_IT_DISABLE`**  
**Description:** Disable the Exti Line falling edge trigger.  
**Parameters:** `_EXTILINE_`: specifies the COMP Exti sources to be disabled. This parameter can be a value of  
**Return value:**None.:
- **`_HAL_COMP_GET_EXTI_LINE`**  
**Description:** Get the specified EXTI line for a comparator instance.  
**Parameters:** `_INSTANCE_`: specifies the COMP instance.  
**Return value:**value: of
- **`_HAL_COMP_EXTI_ENABLE_IT`**  
**Description:** Enable the COMP Exti Line.  
**Parameters:** `_EXTILINE_`: specifies the COMP Exti sources to be enabled. This parameter can be a value of  
**Return value:**None.:
- **`_HAL_COMP_EXTI_DISABLE_IT`**  
**Description:** Disable the COMP Exti Line.  
**Parameters:** `_EXTILINE_`: specifies the COMP Exti sources to be disabled. This parameter can be a value of  
**Return value:**None.:
- **`_HAL_COMP_EXTI_GET_FLAG`**  
**Description:** Checks whether the specified EXTI line flag is set or not.  
**Parameters:** `_FLAG_`: specifies the COMP Exti sources to be checked. This parameter can be a value of  
**Return value:**The state of `_FLAG_` (SET or RESET).
- **`_HAL_COMP_EXTI_CLEAR_FLAG`**  
**Description:** Clear the COMP Exti flags.  
**Parameters:** `_FLAG_`: specifies the COMP Exti sources to be cleared. This parameter can be a value of  
**Return value:**None.:
- **`_HAL_COMP_EXTI_GENERATE_SWIT`**  
**Description:** Generates a Software interrupt on selected EXTI line.  
**Parameters:** `_EXTILINE_`: specifies the COMP Exti sources to trig. This parameter can be a value of  
**Return value:**None:

#### ***COMP ExtiLineEvent***

- **`COMP_EXTI_LINE_COMP1_EVENT`**  
External interrupt line 21 Connected to COMP1
- **`COMP_EXTI_LINE_COMP2_EVENT`**  
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 Private Constants***

- **COMP1\_START\_DELAY\_CPU\_CYCLES**
- **COMP2\_START\_DELAY\_CPU\_CYCLES**
- **COMP\_STATE\_BIT\_LOCK**

#### ***COMP Private Macro***

- **\_\_COMP\_CSR\_CMPXOUT**

**Description:** Select the COMP register CSR bit CMPxOUT corresponding to the selected COMP instance.

**Parameters:** \_\_HANDLE\_\_: COMP handle

**Return value:** Comparator: register CSR bit COMP\_CSR\_CMP1OUT or COMP\_CSR\_CMP2OUT

- **\_\_COMP\_IS\_ENABLED**

**Description:** Verification of COMP state: enabled or disabled.

**Parameters:** \_\_HANDLE\_\_: COMP handle

**Return value:** SET: (COMP enabled) or RESET (COMP disabled)

#### ***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\_DISABLED**

Window mode disabled: COMP1 non-inverting input is independant

- **COMP\_WINDOWMODE\_ENABLED**

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**

## 7 HAL COMP Extension Driver

### 7.1 COMPEx Firmware driver defines

#### 7.1.1 COMPEx

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 PA6 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**

***COMP Private Macro***

- **\_\_COMP\_ROUTING\_INTERFACE\_TOBECONFIGURED**

**Description:** Specifies whether Routing Interface (RI) needs to be configured for switches of comparator non-inverting input.

**Parameters:** \_\_HANDLE\_\_: COMP handle.

**Return value:** None.:

## 8 HAL CORTEX Generic Driver

### 8.1 CORTEX Firmware driver API description

The following section lists the various functions of the CORTEX library.

#### 8.1.1 Initialization and de-initialization functions

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

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

#### 8.1.2 Peripheral Control functions

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

- `HAL_NVIC_GetPriorityGrouping()`
- `HAL_NVIC_GetPriority()`
- `HAL_NVIC_SetPendingIRQ()`
- `HAL_NVIC_GetPendingIRQ()`
- `HAL_NVIC_ClearPendingIRQ()`
- `HAL_NVIC_GetActive()`
- `HAL_SYSTICK_CLKSourceConfig()`
- `HAL_SYSTICK_IRQHandler()`
- `HAL_SYSTICK_Callback()`

#### 8.1.3 `HAL_NVIC_SetPriorityGrouping`

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

|               |                                                                                                                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <p>priority 1 bits for subpriority</p> <ul style="list-style-type: none"> <li>- <b>NVIC_PRIORITYGROUP_4</b> 4 bits for pre-emption</li> </ul> <p>priority 0 bits for subpriority</p>                  |
| Return values | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                             |
| Notes         | <ul style="list-style-type: none"> <li>• 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.</li> </ul> |

### 8.1.4 HAL\_NVIC\_SetPriority

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NVIC_SetPriority ( IRQn_Type IRQn, uint32_t PreemptPriority, uint32_t SubPriority)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Function Description | Sets the priority of an interrupt.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>IRQn</b> : 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))</li> <li>• <b>PreemptPriority</b> : 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</li> <li>• <b>SubPriority</b> : 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.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

### 8.1.5 HAL\_NVIC\_EnableIRQ

|                      |                                                                                                                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NVIC_EnableIRQ ( IRQn_Type IRQn)</b>                                                                                                                                                                                                                             |
| Function Description | Enables a device specific interrupt in the NVIC interrupt controller.                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>IRQn</b> : 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))</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                    |

- |       |                                                                                                                                                              |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Notes | <ul style="list-style-type: none"><li>To configure interrupts priority correctly, the NVIC_PriorityGroupConfig() function should be called before.</li></ul> |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|

### 8.1.6 HAL\_NVIC\_DisableIRQ

|                      |                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NVIC_DisableIRQ ( IRQn_Type IRQn)</b>                                                                                                                                                                                                                          |
| Function Description | Disables a device specific interrupt in the NVIC interrupt controller.                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li><b>IRQn</b> : 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))</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                      |

### 8.1.7 HAL\_NVIC\_SystemReset

|                      |                                                       |
|----------------------|-------------------------------------------------------|
| Function Name        | <b>void HAL_NVIC_SystemReset ( void )</b>             |
| Function Description | Initiates a system reset request to reset the MCU.    |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul> |

### 8.1.8 HAL\_SYSTICK\_Config

|                      |                                                                                                                |
|----------------------|----------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_SYSTICK_Config ( uint32_t TicksNumb)</b>                                                       |
| Function Description | Initializes the System Timer and its interrupt, and starts the System Tick Timer.                              |
| Parameters           | <ul style="list-style-type: none"><li><b>TicksNumb</b> : Specifies the ticks Number of ticks between</li></ul> |

---

|               |                                                                                                                                                                            |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | two interrupts.                                                                                                                                                            |
| Return values | <ul style="list-style-type: none"> <li>• <b>status : - 0 Function succeeded.</b> <ul style="list-style-type: none"> <li>- <b>1 Function failed.</b></li> </ul> </li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                  |

### 8.1.9 HAL\_NVIC\_GetPriorityGrouping

|                      |                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_NVIC_GetPriorityGrouping ( void )</b>                                                                    |
| Function Description | Gets the priority grouping field from the NVIC Interrupt Controller.                                                     |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Priority grouping field (SCB-&gt;AIRCR [10:8] PRIGROUP field)</b></li> </ul> |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                |

### 8.1.10 HAL\_NVIC\_GetPriority

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NVIC_GetPriority ( IRQn_Type IRQn, uint32_t PriorityGroup, uint32_t * pPreemptPriority, uint32_t * pSubPriority )</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Function Description | Gets the priority of an interrupt.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>IRQn :</b> 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))</li> <li>• <b>PriorityGroup :</b> the priority grouping bits length. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>NVIC_PRIORITYGROUP_0</b> 0 bits for pre-emption priority 4 bits for subpriority</li> <li>- <b>NVIC_PRIORITYGROUP_1</b> 1 bits for pre-emption priority 3 bits for subpriority</li> <li>- <b>NVIC_PRIORITYGROUP_2</b> 2 bits for pre-emption priority 2 bits for subpriority</li> <li>- <b>NVIC_PRIORITYGROUP_3</b> 3 bits for pre-emption priority 1 bits for subpriority</li> <li>- <b>NVIC_PRIORITYGROUP_4</b> 4 bits for pre-emption priority 0 bits for subpriority</li> </ul> </li> <li>• <b>pPreemptPriority :</b> Pointer on the Preemptive priority value (starting from 0).</li> </ul> |

---

|               |                                                                                                                               |
|---------------|-------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"> <li>• <b>pSubPriority</b> : Pointer on the Subpriority value (starting from 0).</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                     |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                     |

### 8.1.11 HAL\_NVIC\_SetPendingIRQ

|                      |                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NVIC_SetPendingIRQ ( IRQn_Type IRQn)</b>                                                                                                                                                                                                                           |
| Function Description | Sets Pending bit of an external interrupt.                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>IRQn</b> : 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))</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                      |

### 8.1.12 HAL\_NVIC\_GetPendingIRQ

|                      |                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_NVIC_GetPendingIRQ ( IRQn_Type IRQn)</b>                                                                                                                                                                                                                       |
| 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"> <li>• <b>IRQn</b> : 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))</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>status</b> : - <b>0</b> <b>Interrupt status is not pending.</b> <ul style="list-style-type: none"> <li>– <b>1</b> <b>Interrupt status is pending.</b></li> </ul> </li> </ul>                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                      |

### 8.1.13 HAL\_NVIC\_ClearPendingIRQ

|                      |                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NVIC_ClearPendingIRQ ( IRQn_Type IRQn)</b>                                                                                                                                                                                                                         |
| Function Description | Clears the pending bit of an external interrupt.                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>IRQn</b> : 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))</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                      |

### 8.1.14 HAL\_NVIC\_GetActive

|                      |                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_NVIC_GetActive ( IRQn_Type IRQn)</b>                                                                                                                                                                                                                           |
| Function Description | Gets active interrupt ( reads the active register in NVIC and returns the active bit).                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>IRQn</b> : 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))</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>status</b> : - <b>0</b> Interrupt status is not pending.<br/>– <b>1</b> Interrupt status is pending.</li> </ul>                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                      |

### 8.1.15 HAL\_SYSTICK\_CLKSourceConfig

|                      |                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SYSTICK_CLKSourceConfig ( uint32_t CLKSource)</b>                                                                                                                                                                                                                                                                                                                 |
| Function Description | Configures the SysTick clock source.                                                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>CLKSource</b> : specifies the SysTick clock source. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>SYSTICK_CLKSOURCE_HCLK_DIV8</b> AHB clock divided by 8 selected as SysTick clock source.</li> <li>– <b>SYSTICK_CLKSOURCE_HCLK</b> AHB clock selected as</li> </ul> </li> </ul> |

SysTick clock source.

|               |                                                         |
|---------------|---------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 8.1.16 HAL\_SYSTICK\_IRQHandler

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_SYSTICK_IRQHandler ( void )</b>             |
| Function Description | This function handles SYSTICK interrupt request.        |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 8.1.17 HAL\_SYSTICK\_Callback

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_SYSTICK_Callback ( void )</b>               |
| Function Description | SYSTICK callback.                                       |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

## 8.2 CORTEX Firmware driver defines

### 8.2.1 CORTEX

CORTEX

#### *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**

**CORTEX SysTick clock source**

- **SYSTICK\_CLKSOURCE\_HCLK\_DIV8**
- **SYSTICK\_CLKSOURCE\_HCLK**

**CORTEX SysTick clock source**

- **\_\_HAL\_CORTEX\_SYSTICKCLK\_CONFIG**

**Description:** Configures the SysTick clock source.

**Parameters:** \_\_CLKSRC\_\_: specifies the SysTick clock source. This parameter can be one of the following values: 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 value:**None:

**CORTEX SysTick clock source**

- **IS\_SYSTICK\_CLK\_SOURCE**

## 9 HAL CRC Generic Driver

### 9.1 CRC Firmware driver registers structures

#### 9.1.1 CRC\_HandleTypeDef

*CRC\_HandleTypeDef* is defined in the `stm32l1xx_hal_crc.h`

##### 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

### 9.2 CRC Firmware driver API description

The following section lists the various functions of the CRC library.

#### 9.2.1 How to use this driver

The CRC HAL driver can be used as follows:

1. Enable CRC AHB clock using `__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.

#### 9.2.2 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
- [`HAL\_CRC\_Init\(\)`](#)
- [`HAL\_CRC\_DeInit\(\)`](#)
- [`HAL\_CRC\_MspInit\(\)`](#)
- [`HAL\_CRC\_MspDeInit\(\)`](#)

### 9.2.3 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.
- [\*\*HAL\\_CRC\\_Accumulate\(\)\*\*](#)
- [\*\*HAL\\_CRC\\_Calculate\(\)\*\*](#)

### 9.2.4 Peripheral State functions

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

- [\*\*HAL\\_CRC\\_GetState\(\)\*\*](#)
- [\*\*HAL\\_CRC\\_Accumulate\(\)\*\*](#)
- [\*\*HAL\\_CRC\\_Calculate\(\)\*\*](#)

### 9.2.5 HAL\_CRC\_Init

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

### 9.2.6 HAL\_CRC\_DeInit

|                      |                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_CRC_DeInit ( <a href="#">CRC_HandleTypeDef</a> * hcrc)</b>                                                                                            |
| Function Description | DeInitializes the CRC peripheral.                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hcrc</b> : pointer to a <a href="#">CRC_HandleTypeDef</a> structure that contains the configuration information for CRC</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                      |

### 9.2.7 HAL\_CRC\_MspInit

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_CRC_MspInit ( <i>CRC_HandleTypeDef</i> * hcrc)</b>                                                                                                      |
| Function Description | Initializes the CRC MSP.                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcrc</b> : pointer to a <i>CRC_HandleTypeDef</i> structure that contains the configuration information for CRC</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 9.2.8 HAL\_CRC\_MspDeInit

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_CRC_MspDeInit ( <i>CRC_HandleTypeDef</i> * hcrc)</b>                                                                                                    |
| Function Description | Deinitializes the CRC MSP.                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcrc</b> : pointer to a <i>CRC_HandleTypeDef</i> structure that contains the configuration information for CRC</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 9.2.9 HAL\_CRC\_Accumulate

|                      |                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_CRC_Accumulate ( <i>CRC_HandleTypeDef</i> * hcrc,<br/>                                  uint32_t pBuffer, uint32_t BufferLength)</b>                                                                                                    |
| 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"><li>• <b>hcrc</b> : pointer to a <i>CRC_HandleTypeDef</i> structure that contains the configuration information for CRC</li><li>• <b>pBuffer</b> : pointer to the buffer containing the data to be computed</li></ul> |

- |               |                                                                                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"><li>• <b>BufferLength</b> : length of the buffer to be computed (defined in word, 4 bytes)</li></ul> |
| Return values | <ul style="list-style-type: none"><li>• <b>32-bit CRC</b></li></ul>                                                                    |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                |

### 9.2.10 HAL\_CRC\_Calculate

|                      |                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_CRC_Calculate ( <i>CRC_HandleTypeDef</i> * hcrc,<br/>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"><li>• <b>hcrc</b> : pointer to a <i>CRC_HandleTypeDef</i> structure that contains the configuration information for CRC</li><li>• <b>pBuffer</b> : Pointer to the buffer containing the data to be computed</li><li>• <b>BufferLength</b> : Length of the buffer to be computed (defined in word, 4 bytes)</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>32-bit CRC</b></li></ul>                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                |

### 9.2.11 HAL\_CRC\_GetState

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_CRC_StateTypeDef HAL_CRC_GetState ( <i>CRC_HandleTypeDef</i> * hcrc)</code>                                                                               |
| Function Description | Returns the CRC state.                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcrc</b> : pointer to a <i>CRC_HandleTypeDef</i> structure that contains the configuration information for CRC</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

## 9.2.12 HAL\_CRC\_Accumulate

|                      |                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_CRC_Accumulate ( <i>CRC_HandleTypeDef</i> * hcrc,<br/>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"> <li>• <b>hcrc</b> : pointer to a <i>CRC_HandleTypeDef</i> structure that contains the configuration information for CRC</li> <li>• <b>pBuffer</b> : pointer to the buffer containing the data to be computed</li> <li>• <b>BufferLength</b> : length of the buffer to be computed (defined in word, 4 bytes)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>32-bit CRC</b></li> </ul>                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                  |

## 9.2.13 HAL\_CRC\_Calculate

|                      |                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_CRC_Calculate ( <i>CRC_HandleTypeDef</i> * hcrc,<br/>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"> <li>• <b>hcrc</b> : pointer to a <i>CRC_HandleTypeDef</i> structure that contains the configuration information for CRC</li> <li>• <b>pBuffer</b> : Pointer to the buffer containing the data to be computed</li> <li>• <b>BufferLength</b> : Length of the buffer to be computed (defined in word, 4 bytes)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>32-bit CRC</b></li> </ul>                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                  |

## 9.3 CRC Firmware driver defines

### 9.3.1 CRC

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

## 10 HAL CRYP Generic Driver

### 10.1 CRYP Firmware driver registers structures

#### 10.1.1 CRYP\_InitTypeDef

*CRYP\_InitTypeDef* is defined in the `stm32l1xx_hal_cryp.h`

##### 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

#### 10.1.2 CRYP\_HandleTypeDef

*CRYP\_HandleTypeDef* is defined in the `stm32l1xx_hal_cryp.h`

##### Data Fields

- `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

- `CRYP_InitTypeDef CRYP_HandleTypeDef::Init` CRYP required parameters
- `uint8_t* CRYP_HandleTypeDef::pCrypInBuffPtr` Pointer to CRYP processing (encryption, decryption,...) buffer
- `uint8_t* CRYP_HandleTypeDef::pCrypOutBuffPtr` Pointer to CRYP processing (encryption, decryption,...) buffer
- `__IO uint16_t CRYP_HandleTypeDef::CrypInCount` Counter of inputed data
- `__IO uint16_t CRYP_HandleTypeDef::CrypOutCount` Counter of outputed data
- `HAL_StatusTypeDef CRYP_HandleTypeDef::Status` CRYP peripheral status
- `HAL_PhaseTypeDef CRYP_HandleTypeDef::Phase` CRYP peripheral phase

- **DMA\_HandleTypeDef\* CRYP\_HandleTypeDef::hdmain** CRYP In DMA handle parameters
- **DMA\_HandleTypeDef\* CRYP\_HandleTypeDef::hdmaout** CRYP Out DMA handle parameters
- **HAL\_LockTypeDef CRYP\_HandleTypeDef::Lock** CRYP locking object
- **\_IO HAL\_CRYP\_STATETypeDef CRYP\_HandleTypeDef::State** CRYP peripheral state

## 10.2 CRYP Firmware driver API description

The following section lists the various functions of the CRYP library.

### 10.2.1 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize the CRYP according to the specified parameters in the CRYP\_InitTypeDef and creates the associated handle
- Deinitialize the CRYP peripheral
- Initialize the CRYP MSP
- Deinitialize CRYP MSP
- **HAL\_CRYP\_Init()**
- **HAL\_CRYP\_DelInit()**
- **HAL\_CRYP\_MspInit()**
- **HAL\_CRYP\_MspDelInit()**

### 10.2.2 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
- **HAL\_CRYP\_AESECB\_Encrypt()**
- **HAL\_CRYP\_AESCBC\_Encrypt()**
- **HAL\_CRYP\_AESCTR\_Encrypt()**
- **HAL\_CRYP\_AESECB\_Decrypt()**
- **HAL\_CRYP\_AESCBC\_Decrypt()**
- **HAL\_CRYP\_AESCTR\_Decrypt()**
- **HAL\_CRYP\_AESECB\_Encrypt\_IT()**
- **HAL\_CRYP\_AESCBC\_Encrypt\_IT()**
- **HAL\_CRYP\_AESCTR\_Encrypt\_IT()**
- **HAL\_CRYP\_AESECB\_Decrypt\_IT()**
- **HAL\_CRYP\_AESCBC\_Decrypt\_IT()**
- **HAL\_CRYP\_AESCTR\_Decrypt\_IT()**
- **HAL\_CRYP\_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\(\)\*](#)

### 10.2.3 DMA callback functions

This section provides DMA callback functions:

- DMA Input data transfer complete
- DMA Output data transfer complete
- DMA error
- [\*HAL\\_CRYP\\_ErrorCallback\(\)\*](#)
- [\*HAL\\_CRYP\\_InCpltCallback\(\)\*](#)
- [\*HAL\\_CRYP\\_OutCpltCallback\(\)\*](#)

### 10.2.4 CRYP IRQ handler management

This section provides CRYP IRQ handler function.

- [\*HAL\\_CRYP\\_IRQHandler\(\)\*](#)

### 10.2.5 Peripheral State functions

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

- [\*HAL\\_CRYP\\_GetState\(\)\*](#)

### 10.2.6 HAL\_CRYP\_Init

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_CRYP_Init ( <a href="#">CRYP_HandleTypeDef</a> * hcryp)</code>                                                                                            |
| Function Description | Initializes the CRYP according to the specified parameters in the <code>CRYP_InitTypeDef</code> and creates the associated handle.                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hcryp</b> : pointer to a <code>CRYP_HandleTypeDef</code> structure that contains the configuration information for CRYP module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                             |

### 10.2.7 HAL\_CRYP\_DelInit

---

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_CRYP_DeInit (</b><br><b>CRYP_HandleTypeDef * hcryp)</b>                                                                                       |
| Function Description | Deinitializes the CRYP peripheral.                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |

### 10.2.8 HAL\_CRYP\_MspInit

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_CRYP_MspInit (</b><br><b>CRYP_HandleTypeDef * hcryp)</b>                                                                                                   |
| Function Description | Initializes the CRYP MSP.                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |

### 10.2.9 HAL\_CRYP\_MspDeInit

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_CRYP_MspDeInit (</b><br><b>CRYP_HandleTypeDef * hcryp)</b>                                                                                                 |
| Function Description | Deinitializes CRYP MSP.                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |

### 10.2.10 HAL\_CRYP\_AESECB\_Encrypt

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

### 10.2.11 HAL\_CRYP\_AESCBC\_Encrypt

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

### 10.2.12 HAL\_CRYP\_AESCTR\_Encrypt

|               |                                                                                                                                     |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_CRYP_AESCTR_Encrypt (</code><br><code>CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t</code> |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------|

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

### 10.2.13 HAL\_CRYP\_AESECB\_Decrypt

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

### 10.2.14 HAL\_CRYP\_AESCBC\_Decrypt

|               |                                                                                                                                                                                  |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_CRYP_AESCBC_Decrypt (</b><br><b>CRYP_HandleTypeDef * hcryp, uint8_t * pCypherData,</b><br><b>uint16_t Size, uint8_t * pPlainData, uint32_t Timeout)</b> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Initializes the CRYP peripheral in AES ECB decryption mode then decrypted pCypherData.                                                                                                                                                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li><b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module</li> <li><b>pCypherData</b> : Pointer to the ciphertext buffer (aligned on u32)</li> <li><b>Size</b> : Length of the plaintext buffer, must be a multiple of 16.</li> <li><b>pPlainData</b> : Pointer to the plaintext buffer (aligned on u32)</li> <li><b>Timeout</b> : Specify Timeout value</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                               |

### 10.2.15 HAL\_CRYP\_AESCTR\_Decrypt

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

### 10.2.16 HAL\_CRYP\_AESECB\_Encrypt\_IT

|                      |                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_CRYP_AESECB_Encrypt_IT (</b><br><b>CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t</b><br><b>Size, uint8_t * pCypherData)</b> |
| Function Description | Initializes the CRYP peripheral in AES ECB encryption mode using Interrupt.                                                                                       |

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

### 10.2.17 HAL\_CRYP\_AESCBC\_Encrypt\_IT

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

### 10.2.18 HAL\_CRYP\_AESCTR\_Encrypt\_IT

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_CRYP_AESCTR_Encrypt_IT (</b><br><b>CRYP_HandleTypeDef * hcryp, uint8_t * pPlainData, uint16_t</b><br><b>Size, uint8_t * pCypherData)</b>      |
| Function Description | Initializes the CRYP peripheral in AES CTR encryption mode using Interrupt.                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li><b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module</li> </ul> |

---

|               |                                                                                                                                                                                                                                                                                                    |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"> <li>• <b>pPlainData</b> : Pointer to the plaintext buffer (aligned on u32)</li> <li>• <b>Size</b> : Length of the plaintext buffer, must be a multiple of 16 bytes</li> <li>• <b>pCypherData</b> : Pointer to the ciphertext buffer (aligned on u32)</li> </ul> |
| Return values | • <b>HAL status</b>                                                                                                                                                                                                                                                                                |
| Notes         | • None.                                                                                                                                                                                                                                                                                            |

### 10.2.19 HAL\_CRYP\_AESECB\_Decrypt\_IT

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

### 10.2.20 HAL\_CRYP\_AESCBC\_Decrypt\_IT

|                      |                                                                                                                                                                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_CRYP_AESCBC_Decrypt_IT (<br/>    <b>CRYP_HandleTypeDef</b> * hcryp, uint8_t * pCypherData,<br/>    uint16_t Size, uint8_t * pPlainData)</code>                                                                                                                                                                   |
| Function Description | Initializes the CRYP peripheral in AES CBC decryption mode using IT.                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module</li> <li>• <b>pCypherData</b> : Pointer to the ciphertext buffer (aligned on u32)</li> <li>• <b>Size</b> : Length of the plaintext buffer, must be a multiple of 16</li> </ul> |

- |               |                                                                                                                          |
|---------------|--------------------------------------------------------------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"> <li>• <b>pPlainData</b> : Pointer to the plaintext buffer (aligned on u32)</li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> <li>• None.</li> </ul>                                   |

### 10.2.21 HAL\_CRYP\_AESCTR\_Decrypt\_IT

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

### 10.2.22 HAL\_CRYP\_AESECB\_Encrypt\_DMA

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

## Notes

- None.

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

### 10.2.24 HAL\_CRYP\_AESCTR\_Encrypt\_DMA

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

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

### 10.2.26 HAL\_CRYP\_AESCBC\_Decrypt\_DMA

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

## 10.2.27 HAL\_CRYP\_AESCTR\_Decrypt\_DMA

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

## 10.2.28 HAL\_CRYP\_ErrorCallback

|                      |                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_CRYP_ErrorCallback (CRYP_HandleTypeDef * hcryp)</code>                                                                                                    |
| Function Description | CRYP error callback.                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module</li> </ul> |
| Return values        | • None.                                                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                |

## 10.2.29 HAL\_CRYP\_InCpltCallback

|                      |                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_CRYP_InCpltCallback (CRYP_HandleTypeDef * hcryp)</code>                                            |
| Function Description | Input transfer completed callback.                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that</li> </ul> |

contains the configuration information for CRYP module

|               |                                                       |
|---------------|-------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>None.</li></ul> |
| Notes         | <ul style="list-style-type: none"><li>None.</li></ul> |

### 10.2.30 HAL\_CRYP\_OutCpltCallback

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

### 10.2.31 HAL\_CRYP\_IRQHandler

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

### 10.2.32 HAL\_CRYP\_GetState

|               |                                                                                     |
|---------------|-------------------------------------------------------------------------------------|
| Function Name | <b>HAL_CRYP_STATETypeDef HAL_CRYP_GetState ( <i>CRYP_HandleTypeDef</i> * hcryp)</b> |
|---------------|-------------------------------------------------------------------------------------|

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

## 10.3 CRYP Firmware driver defines

### 10.3.1 CRYP

CRYP

#### *AES Clear Flags*

- **AES\_CLEARFLAG\_CCF**  
Computation Complete Flag Clear
- **AES\_CLEARFLAG\_RDERR**  
Read Error Clear
- **AES\_CLEARFLAG\_WRERR**  
Write Error Clear

#### *AES Flags*

- **AES\_FLAG\_CCF**  
Computation Complete Flag
- **AES\_FLAG\_RDERR**  
Read Error Flag
- **AES\_FLAG\_WRERR**  
Write Error Flag

#### *AES Interrupts*

- **AES\_IT\_CC**  
Computation Complete interrupt
- **AES\_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***

- **\_HAL\_CRYP\_RESET\_HANDLE\_STATE**

**Description:** Reset CRYP handle state.

**Parameters:** `_HANDLE_`: specifies the CRYP Handle.

**Return value:**None

- **\_HAL\_CRYP\_ENABLE**

**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:** `_MODE_`: The algorithm mode.

**Return value:**None

- **\_HAL\_CRYP\_GET\_FLAG**

**Description:** Check whether the specified CRYP flag is set or not.

**Parameters:** `_FLAG_`: specifies the flag to check. This parameter can be one of the following values: AES\_FLAG\_CCF : Computation Complete Flag AES\_FLAG\_RDERR : Read Error Flag AES\_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: AES\_CLEARFLAG\_CCF : Computation Complete Clear Flag AES\_CLEARFLAG\_RDERR : Read Error Clear AES\_CLEARFLAG\_WRERR : Write Error Clear

**Return value:**None

- **\_HAL\_CRYP\_ENABLE\_IT**

**Description:** Enable the CRYP interrupt.

**Parameters:** `_INTERRUPT_`: CRYP Interrupt.

**Return value:**None

- **\_HAL\_CRYP\_DISABLE\_IT**

**Description:** Disable the CRYP interrupt.

**Parameters:** `_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_`: CRYP handle `_INTERRUPT_`: CRYP interrupt source to check This parameter can be one of the following values: AES\_IT\_CC : Computation Complete interrupt AES\_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: AES\_CLEARFLAG\_CCF : Computation Complete Clear Flag AES\_CLEARFLAG\_RDERR : Read Error Clear AES\_CLEARFLAG\_WRERR : Write Error Clear

**Return value:**None

***CRYP Private Defines***

- **CRYP\_ALGO\_CHAIN\_MASK**

## 11 HAL CRYP Extension Driver

### 11.1 CRYPEx Firmware driver API description

The following section lists the various functions of the CRYPEx library.

#### 11.1.1 Extended features functions

This section provides callback functions:

- Computation completed.
- [\*\*HAL\\_CRYPEx\\_ComputationCpltCallback\(\)\*\*](#)

#### 11.1.2 **HAL\_CRYPEx\_ComputationCpltCallback**

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_CRYPEx_ComputationCpltCallback (</b><br><b>CRYP_HandleTypeDef * hcryp)</b>                                                                                 |
| Function Description | Computation completed callbacks.                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hcryp</b> : pointer to a CRYP_HandleTypeDef structure that contains the configuration information for CRYP module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |

### 11.2 CRYPEx Firmware driver defines

#### 11.2.1 CRYPEx

CRYPEx

## 12 HAL DAC Generic Driver

### 12.1 DAC Firmware driver registers structures

#### 12.1.1 DAC\_HandleTypeDef

*DAC\_HandleTypeDef* is defined in the `stm32l1xx_hal_dac.h`

##### 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

#### 12.1.2 DAC\_ChannelConfTypeDef

*DAC\_ChannelConfTypeDef* is defined in the `stm32l1xx_hal_dac.h`

##### 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\*](#)

### 12.2 DAC Firmware driver API description

The following section lists the various functions of the DAC library.

## 12.2.1 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\_Pin9) using DAC\_Trigger\_Ext\_IT9. The used pin (GPIOx\_Pin9) 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 wave generation feature

Both DAC channels can be used to generate

1. Noise wave
2. Triangle wave

### 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 correspondence

The analog output voltage on each DAC channel pin is determined by the following equation:  $DAC_{OUTx} = VREF+ * DOR / 4095$  with DOR is the Data Output Register VEF+ 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:

1. DAC channel1 : mapped on DMA1 channel2 which must be already configured
2. 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

## 12.2.2 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 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\_DAC\_ConvHalfCpltCallbackCh1() or HAL\_DAC\_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() function is executed and user can add his own code by customization of function pointer HAL\_DAC\_ErrorCallbackCh1
- In case of DMA underrun, DAC interruption triggers and execute internal function HAL\_DAC\_IRQHandler. HAL\_DAC\_DMADMAUnderrunCallbackCh1() or HAL\_DAC\_DMADMAUnderrunCallbackCh2() function is executed and user can add his own code by customization of function pointer HAL\_DAC\_DMADMAUnderrunCallbackCh1 or HAL\_DAC\_DMADMAUnderrunCallbackCh2 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

### 12.2.3 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize and configure the DAC.
- De-initialize the DAC.
- `HAL_DAC_Init()`
- `HAL_DAC_DelInit()`
- `HAL_DAC_MspInit()`
- `HAL_DAC_MspDelInit()`

### 12.2.4 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.
- `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()`

### 12.2.5 Peripheral Control functions

This section provides functions allowing to:

- Configure channels.
- Set the specified data holding register value for DAC channel.
- [\*\*HAL\\_DAC\\_ConfigChannel\(\)\*\*](#)
- [\*\*HAL\\_DAC\\_SetValue\(\)\*\*](#)

### 12.2.6 Peripheral State and Errors functions

This subsection provides functions allowing to

- Check the DAC state.
- Check the DAC Errors.
- [\*\*HAL\\_DAC\\_GetState\(\)\*\*](#)
- [\*\*HAL\\_DAC\\_GetError\(\)\*\*](#)

### 12.2.7 HAL\_DAC\_Init

|                      |                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_DAC_Init ( <a href="#">DAC_HandleTypeDef</a> * hdac)</b>                                                                                           |
| Function Description | Initializes the DAC peripheral according to the specified parameters in the DAC_InitStruct.                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                         |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |

### 12.2.8 HAL\_DAC\_DeInit

|                      |                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_DAC_DeInit ( <a href="#">DAC_HandleTypeDef</a> * hdac)</b>                                                                                         |
| Function Description | Deinitializes the DAC peripheral registers to their default reset values.                                                                                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                         |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |

## 12.2.9 HAL\_DAC\_MspInit

|                      |                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_DAC_MspInit ( <i>DAC_HandleTypeDef</i> * hdac)</b>                                                                                                                |
| Function Description | Initializes the DAC MSP.                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                     |

## 12.2.10 HAL\_DAC\_MspDeInit

|                      |                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_DAC_MspDeInit ( <i>DAC_HandleTypeDef</i> * hdac)</b>                                                                                                              |
| Function Description | Deinitializes the DAC MSP.                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                     |

## 12.2.11 HAL\_DAC\_Start

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_DAC_Start ( <i>DAC_HandleTypeDef</i> * hdac, uint32_t Channel)</b>                                                                                                                                                                                                                                                                                                                                                      |
| Function Description | Enables DAC and starts conversion of channel.                                                                                                                                                                                                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> <li>• <b>Channel</b> : The selected DAC channel. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>DAC_CHANNEL_1</b> : DAC Channel1 selected</li> <li>– <b>DAC_CHANNEL_2</b> : DAC Channel2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                        |

### 12.2.12 HAL\_DAC\_Stop

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

### 12.2.13 HAL\_DAC\_Start\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_DAC_Start_DMA ( <i>DAC_HandleTypeDef</i> * hdac, uint32_t Channel, uint32_t * pData, uint32_t Length, uint32_t Alignment)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Enables DAC and starts conversion of channel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a <i>DAC_HandleTypeDef</i> structure that contains the configuration information for the specified DAC.</li> <li>• <b>Channel</b> : The selected DAC channel. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>DAC_CHANNEL_1</b> : DAC Channel1 selected</li> <li>– <b>DAC_CHANNEL_2</b> : DAC Channel2 selected</li> </ul> </li> <li>• <b>pData</b> : The destination peripheral Buffer address.</li> <li>• <b>Length</b> : The length of data to be transferred from memory to DAC peripheral</li> <li>• <b>Alignment</b> : Specifies the data alignment for DAC channel. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>DAC_ALIGN_8B_R</b> : 8bit right data alignment selected</li> <li>– <b>DAC_ALIGN_12B_L</b> : 12bit left data alignment selected</li> <li>– <b>DAC_ALIGN_12B_R</b> : 12bit right data alignment selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

- 
- |       |                                                         |
|-------|---------------------------------------------------------|
| Notes | <ul style="list-style-type: none"> <li>None.</li> </ul> |
|-------|---------------------------------------------------------|

### 12.2.14 HAL\_DAC\_Stop\_DMA

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

### 12.2.15 HAL\_DAC\_GetValue

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_DAC_GetValue ( DAC_HandleTypeDef * hdac, uint32_t Channel)</b>                                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Returns the last data output value of the selected DAC channel.                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li><b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> <li><b>Channel</b> : The selected DAC channel. This parameter can be one of the following values:               <ul style="list-style-type: none"> <li>– <b>DAC_CHANNEL_1</b> : DAC Channel1 selected</li> <li>– <b>DAC_CHANNEL_2</b> : DAC Channel2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>The selected DAC channel data output value.</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                    |

### 12.2.16 HAL\_DAC\_IRQHandler

|                      |                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_DAC_IRQHandler ( <i>DAC_HandleTypeDef</i> * hdac)</b>                                                                                                           |
| Function Description | Handles DAC interrupt request.                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |

### 12.2.17 HAL\_DAC\_ConvCpltCallbackCh1

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

### 12.2.18 HAL\_DAC\_ConvHalfCpltCallbackCh1

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

### 12.2.19 HAL\_DAC\_ErrorCallbackCh1

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_DAC_ErrorCallbackCh1 ( <i>DAC_HandleTypeDef</i> * hdac)</code>                                                                                                        |
| Function Description | Error DAC callback for Channel1.                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a <i>DAC_HandleTypeDef</i> structure that contains the configuration information for the specified DAC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                            |

### 12.2.20 HAL\_DAC\_DMADebugCallbackCh1

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_DAC_DMADebugCallbackCh1 ( <i>DAC_HandleTypeDef</i> * hdac)</code>                                                                                                     |
| Function Description | DMA debug DAC callback for channel1.                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a <i>DAC_HandleTypeDef</i> structure that contains the configuration information for the specified DAC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                            |

### 12.2.21 HAL\_DAC\_SetValue

|                      |                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_DAC_SetValue ( <i>DAC_HandleTypeDef</i> * hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data)</code>                                                                                                                                                                                                                                                         |
| Function Description | Set the specified data holding register value for DAC channel.                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a <i>DAC_HandleTypeDef</i> structure that contains the configuration information for the specified DAC.</li> <li>• <b>Channel</b> : The selected DAC channel. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>DAC_CHANNEL_1</b> : DAC Channel1 selected</li> </ul> </li> </ul> |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"> <li>- <b>DAC_CHANNEL_2</b> : DAC Channel2 selected</li> </ul>                                                                                                                                                                                                                                                                                                                                 |
|               | <ul style="list-style-type: none"> <li>• <b>Alignment</b> : Specifies the data alignment. This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>- <b>DAC_ALIGN_8B_R</b> : 8bit right data alignment selected</li> <li>- <b>DAC_ALIGN_12B_L</b> : 12bit left data alignment selected</li> <li>- <b>DAC_ALIGN_12B_R</b> : 12bit right data alignment selected</li> </ul> </li> </ul> |
|               | <ul style="list-style-type: none"> <li>• <b>Data</b> : Data to be loaded in the selected data holding register.</li> </ul>                                                                                                                                                                                                                                                                                                       |
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                            |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                        |

### 12.2.22 HAL\_DAC\_ConfigChannel

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

### 12.2.23 HAL\_DAC\_GetState

|                      |                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_DAC_StateTypeDef HAL_DAC_GetState (</b><br><b>DAC_HandleTypeDef * hdac)</b>                                                                                            |
| Function Description | return the DAC state                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> </ul> |

---

|               |                                                                    |
|---------------|--------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"> <li><b>HAL state</b></li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>None.</li> </ul>            |

## 12.2.24 HAL\_DAC\_GetError

|                      |                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_DAC_GetError ( <i>DAC_HandleTypeDef</i> * hdac)</b>                                                                                                                |
| Function Description | Return the DAC error code.                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li><b>hdac</b> : pointer to a <i>DAC_HandleTypeDef</i> structure that contains the configuration information for the specified DAC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>DAC Error Code</b></li> </ul>                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                            |

## 12.2.25 HAL\_DAC\_ConfigChannel

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

## 12.2.26 HAL\_DAC\_SetValue

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

### 12.2.27 HAL\_DAC\_GetState

|                      |                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_DAC_StateTypeDef HAL_DAC_GetState (</code><br><code>DAC_HandleTypeDef * hdac)</code>                                                                                |
| Function Description | return the DAC state                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>                                                                                                          |

### 12.2.28 HAL\_DAC\_GetError

|               |                                                                    |
|---------------|--------------------------------------------------------------------|
| Function Name | <code>uint32_t HAL_DAC_GetError ( DAC_HandleTypeDef * hdac)</code> |
|---------------|--------------------------------------------------------------------|

|                      |                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Return the DAC error code.                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>DAC Error Code</b></li> </ul>                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                     |

## 12.3 DAC Firmware driver defines

### 12.3.1 DAC

DAC

*DAC Channel selection*

- **DAC\_CHANNEL\_1**
- **DAC\_CHANNEL\_2**
- **IS\_DAC\_CHANNEL**

*DAC data*

- **IS\_DAC\_DATA**

*DAC data alignment*

- **DAC\_ALIGN\_12B\_R**
- **DAC\_ALIGN\_12B\_L**
- **DAC\_ALIGN\_8B\_R**
- **IS\_DAC\_ALIGN**

*DAC Error Code*

- **HAL\_DAC\_ERROR\_NONE**  
No error
- **HAL\_DAC\_ERROR\_DMAUNDERRUNCH1**  
DAC channel1 DAM underrun error
- **HAL\_DAC\_ERROR\_DMAUNDERRUNCH2**  
DAC channel2 DAM 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\_DHR12R1\_ALIGNEMENT

**Description:** Set DHR12R1 alignment.

**Parameters:** \_\_ALIGNEMENT\_\_: specifies the DAC alignment

**Return value:**None

- \_\_HAL\_DHR12R2\_ALIGNEMENT

**Description:** Set DHR12R2 alignment.

**Parameters:** \_\_ALIGNEMENT\_\_: specifies the DAC alignment

**Return value:**None

- \_\_HAL\_DHR12RD\_ALIGNEMENT

**Description:** Set DHR12RD alignment.

**Parameters:** \_\_ALIGNEMENT\_\_: specifies the DAC alignment

**Return value:**None

- \_\_HAL\_DAC\_ENABLE\_IT

**Description:** Enable the DAC interrupt.

**Parameters:** \_\_HANDLE\_\_: specifies the DAC handle \_\_INTERRUPT\_\_: specifies the DAC interrupt.

**Return value:**None

- \_\_HAL\_DAC\_DISABLE\_IT

**Description:** Disable the DAC interrupt.

**Parameters:** \_\_HANDLE\_\_: specifies the DAC handle \_\_INTERRUPT\_\_: specifies the DAC interrupt.

**Return value:**None

- \_\_HAL\_DAC\_GET\_FLAG

**Description:** Get the selected DAC's flag status.

**Parameters:** \_\_HANDLE\_\_: specifies the DAC handle. \_\_FLAG\_\_: specifies the FLAG.

**Return value:**None

- \_\_HAL\_DAC\_CLEAR\_FLAG

**Description:** Clear the DAC's flag.

**Parameters:** \_\_HANDLE\_\_: specifies the DAC handle. \_\_FLAG\_\_: specifies the 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
- IS\_DAC\_OUTPUT\_BUFFER\_STATE

#### 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
- **IS\_DAC\_TRIGGER**

## 13 HAL DAC Extension Driver

### 13.1 DACEx Firmware driver API description

The following section lists the various functions of the DACEx library.

#### 13.1.1 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.

#### 13.1.2 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.
- [\*\*HAL\\_DACEx\\_DualGetValue\(\)\*\*](#)
- [\*\*HAL\\_DACEx\\_TriangleWaveGenerate\(\)\*\*](#)
- [\*\*HAL\\_DACEx\\_NoiseWaveGenerate\(\)\*\*](#)
- [\*\*HAL\\_DACEx\\_DualSetValue\(\)\*\*](#)
- [\*\*HAL\\_DACEx\\_ConvCpltCallbackCh2\(\)\*\*](#)
- [\*\*HAL\\_DACEx\\_ConvHalfCpltCallbackCh2\(\)\*\*](#)
- [\*\*HAL\\_DACEx\\_ErrorCallbackCh2\(\)\*\*](#)
- [\*\*HAL\\_DACEx\\_DMAUnderrunCallbackCh2\(\)\*\*](#)

#### 13.1.3 HAL\_DACEx\_DualGetValue

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_DACEx_DualGetValue ( <a href="#">DAC_HandleTypeDef</a> * hdac)</code>                                                                                                 |
| Function Description | Returns the last data output value of the selected DAC channel.                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdac</b> : pointer to a <code>DAC_HandleTypeDef</code> structure that contains the configuration information for the specified DAC.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>The selected DAC channel data output value.</b></li></ul>                                                                                     |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

### 13.1.4 HAL\_DACEx\_TriangleWaveGenerate

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_DACEx_TriangleWaveGenerate (</code><br><code>DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t</code><br><code>Amplitude)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Function Description | Enables or disables the selected DAC channel wave generation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> <li>• <b>Channel</b> : The selected DAC channel. This parameter can be one of the following values: DAC_CHANNEL_1 / DAC_CHANNEL_2</li> <li>• <b>Amplitude</b> : Select max triangle amplitude. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>DAC_TRIANGLEAMPLITUDE_1</b> : Select max triangle amplitude of 1</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_3</b> : Select max triangle amplitude of 3</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_7</b> : Select max triangle amplitude of 7</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_15</b> : Select max triangle amplitude of 15</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_31</b> : Select max triangle amplitude of 31</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_63</b> : Select max triangle amplitude of 63</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_127</b> : Select max triangle amplitude of 127</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_255</b> : Select max triangle amplitude of 255</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_511</b> : Select max triangle amplitude of 511</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_1023</b> : Select max triangle amplitude of 1023</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_2047</b> : Select max triangle amplitude of 2047</li> <li>– <b>DAC_TRIANGLEAMPLITUDE_4095</b> : Select max triangle amplitude of 4095</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

### 13.1.5 HAL\_DACEx\_NoiseWaveGenerate

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_DACEx_NoiseWaveGenerate (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t Amplitude)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Enables or disables the selected DAC channel wave generation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> <li>• <b>Channel</b> : The selected DAC channel. This parameter can be one of the following values: DAC_CHANNEL_1 / DAC_CHANNEL_2</li> <li>• <b>Amplitude</b> : Unmask DAC channel LFSR for noise wave generation. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>DAC_LFSRUNMASK_BIT0</b> : Unmask DAC channel LFSR bit0 for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS1_0</b> : Unmask DAC channel LFSR bit[1:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS2_0</b> : Unmask DAC channel LFSR bit[2:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS3_0</b> : Unmask DAC channel LFSR bit[3:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS4_0</b> : Unmask DAC channel LFSR bit[4:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS5_0</b> : Unmask DAC channel LFSR bit[5:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS6_0</b> : Unmask DAC channel LFSR bit[6:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS7_0</b> : Unmask DAC channel LFSR bit[7:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS8_0</b> : Unmask DAC channel LFSR bit[8:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS9_0</b> : Unmask DAC channel LFSR bit[9:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS10_0</b> : Unmask DAC channel LFSR bit[10:0] for noise wave generation</li> <li>– <b>DAC_LFSRUNMASK_BITS11_0</b> : Unmask DAC channel LFSR bit[11:0] for noise wave generation</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

### 13.1.6 HAL\_DACEx\_DualSetValue

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_DACEx_DualSetValue (</code><br><code>DAC_HandleTypeDef * hdac, uint32_t Alignment, uint32_t</code><br><code>Data1, uint32_t Data2)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Set the specified data holding register value for dual DAC channel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> <li>• <b>Alignment</b> : Specifies the data alignment for dual channel DAC. This parameter can be one of the following values:<br/>DAC_ALIGN_8B_R: 8bit right data alignment selected<br/>DAC_ALIGN_12B_L: 12bit left data alignment selected<br/>DAC_ALIGN_12B_R: 12bit right data alignment selected</li> <li>• <b>Data1</b> : Data for DAC Channel2 to be loaded in the selected data holding register.</li> <li>• <b>Data2</b> : Data for DAC Channel1 to be loaded in the selected data holding register.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Notes                | • In dual mode, a unique register access is required to write in both DAC channels at the same time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

### 13.1.7 HAL\_DACEx\_ConvCpltCallbackCh2

|                      |                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_DACEx_ConvCpltCallbackCh2 (</code><br><code>DAC_HandleTypeDef * hdac)</code>                                                                                   |
| Function Description | Conversion complete callback in non blocking mode for Channel2.                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li> </ul> |
| Return values        | • None.                                                                                                                                                                       |
| Notes                | • None.                                                                                                                                                                       |

### 13.1.8 HAL\_DACEx\_ConvHalfCpltCallbackCh2

|                      |                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_DACEx_ConvHalfCpltCallbackCh2 (</code><br><code>DAC_HandleTypeDef * hdac)</code> |
| Function Description | Conversion half DMA transfer callback in non blocking mode for                                  |

Channel2.

|               |                                                                                                                                                                             |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"><li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li></ul> |
| Return values | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |

### 13.1.9 HAL\_DACEx\_ErrorCallbackCh2

|                      |                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_DACEx_ErrorCallbackCh2 ( DAC_HandleTypeDef * hdac)</b>                                                                                                          |
| Function Description | Error DAC callback for Channel2.                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |

### 13.1.10 HAL\_DACEx\_DMAUnderrunCallbackCh2

|                      |                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_DACEx_DMAUnderrunCallbackCh2 ( DAC_HandleTypeDef * hdac)</b>                                                                                                    |
| Function Description | DMA underrun DAC callback for channel2.                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdac</b> : pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                     |

## 13.2 DACEx Firmware driver defines

### 13.2.1 DACEx

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\_WAVEGENERATION\_NONE**
- **DAC\_WAVEGENERATION\_NOISE**
- **DAC\_WAVEGENERATION\_TRIANGLE**
- **IS\_DAC\_GENERATE\_WAVE**
- **DAC\_WAVE\_NOISE**
- **DAC\_WAVE\_TRIANGLE**
- **IS\_DAC\_WAVE**

## 14 HAL DMA Generic Driver

### 14.1 DMA Firmware driver registers structures

#### 14.1.1 DMA\_InitTypeDef

*DMA\_InitTypeDef* is defined in the `stm32l1xx_hal_dma.h`

##### 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\*](#)

#### 14.1.2 DMA\_HandleTypeDef

*DMA\_HandleTypeDef* is defined in the `stm32l1xx_hal_dma.h`

##### Data Fields

- *DMA\_Channel\_TypeDef \* Instance*
- *DMA\_InitTypeDef Init*
- *HAL\_LockTypeDef Lock*
- *HAL\_DMA\_StateTypeDef State*
- *void \* Parent*
- *void(\* XferCpltCallback*

- `void(* XferHalfCpltCallback`
- `void(* XferErrorCallback`
- `__IO uint32_t ErrorCode`

#### 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
- `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
- `__IO uint32_t __DMA_HandleTypeDef::ErrorCode` DMA Error code

## 14.2 DMA Firmware driver API description

The following section lists the various functions of the DMA library.

### 14.2.1 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 Reference manual for connection between peripherals and DMA requests .
2. For a given Channel, program the required configuration through the following parameters: 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\_DMAy\_Channelx\_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

### 14.2.2 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.

- [\*\*HAL\\_DMA\\_Init\(\)\*\*](#)
- [\*\*HAL\\_DMA\\_DeInit\(\)\*\*](#)

### 14.2.3 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
- [\*\*HAL\\_DMA\\_Start\(\)\*\*](#)
- [\*\*HAL\\_DMA\\_Start\\_IT\(\)\*\*](#)
- [\*\*HAL\\_DMA\\_Abort\(\)\*\*](#)
- [\*\*HAL\\_DMA\\_PollForTransfer\(\)\*\*](#)
- [\*\*HAL\\_DMA\\_IRQHandler\(\)\*\*](#)

#### 14.2.4 State and Errors functions

This subsection provides functions allowing to

- Check the DMA state
- Get error code
- **`HAL_DMA_GetState()`**
- **`HAL_DMA_GetError()`**

#### 14.2.5 HAL\_DMA\_Init

|                      |                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b><code>HAL_StatusTypeDef HAL_DMA_Init ( DMA_HandleTypeDef *hdma)</code></b>                                                                                                                    |
| Function Description | Initializes the DMA according to the specified parameters in the <code>DMA_InitTypeDef</code> and create the associated handle.                                                                  |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdma</b> : Pointer to a <code>DMA_HandleTypeDef</code> structure that contains the configuration information for the specified DMA Channel.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                          |

#### 14.2.6 HAL\_DMA\_DeInit

|                      |                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b><code>HAL_StatusTypeDef HAL_DMA_DeInit ( DMA_HandleTypeDef * hdma)</code></b>                                                                                                                 |
| Function Description | Deinitializes the DMA peripheral.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdma</b> : pointer to a <code>DMA_HandleTypeDef</code> structure that contains the configuration information for the specified DMA Channel.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                          |

#### 14.2.7 HAL\_DMA\_Start

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_DMA_Start ( DMA_HandleTypeDef * hdma, uint32_t SrcAddress, uint32_t DstAddress, uint32_t DataLength)</b>                                                                                                                                                                                                                                                                                 |
| Function Description | Starts the DMA Transfer.                                                                                                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li><b>hdma</b> : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.</li> <li><b>SrcAddress</b> : The source memory Buffer address</li> <li><b>DstAddress</b> : The destination memory Buffer address</li> <li><b>DataLength</b> : The length of data to be transferred from source to destination</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                           |

#### 14.2.8 HAL\_DMA\_Start\_IT

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

#### 14.2.9 HAL\_DMA\_Abort

|                      |                                                                    |
|----------------------|--------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_DMA_Abort ( DMA_HandleTypeDef * hdma)</b> |
| Function Description | Aborts the DMA Transfer.                                           |

|               |                                                                                                                                                                                                                                                                                                                                                                 |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li><b>hdma</b> : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.</li> </ul>                                                                                                                                                                             |
| Return values | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                             |
| Notes         | <ul style="list-style-type: none"> <li>After disabling a DMA Channel, a check for wait until the DMA Channel is effectively disabled is added. If a Channel is disabled while a data transfer is ongoing, the current data will be transferred and the Channel will be effectively disabled only after the transfer of this single data is finished.</li> </ul> |

#### 14.2.10 HAL\_DMA\_PollForTransfer

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

#### 14.2.11 HAL\_DMA\_IRQHandler

|                      |                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_DMA_IRQHandler ( DMA_HandleTypeDef * hdma )</code>                                                                                                                   |
| Function Description | Handles DMA interrupt request.                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li><b>hdma</b> : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                             |

### 14.2.12 HAL\_DMA\_GetState

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_DMA_StateTypeDef HAL_DMA_GetState ( DMA_HandleTypeDef * hdma)</b>                                                                                                              |
| Function Description | Returns the DMA state.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdma</b> : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                             |

### 14.2.13 HAL\_DMA\_GetError

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_DMA_GetError ( DMA_HandleTypeDef * hdma)</b>                                                                                                                          |
| Function Description | Return the DMA error code.                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdma</b> : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>DMA Error Code</b></li> </ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                             |

## 14.3 DMA Firmware driver defines

### 14.3.1 DMA

DMA

**DMA\_Data\_buffer\_size**

- **IS\_DMA\_BUFFER\_SIZE**

**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
- **IS\_DMA\_DIRECTION**

**DMA\_Error\_Code**

- **HAL\_DMA\_ERROR\_NONE**  
No error
- **HAL\_DMA\_ERROR\_TE**  
Transfer error
- **HAL\_DMA\_ERROR\_TIMEOUT**  
Timeout error

**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\_ENABLE\_IT**  
**Description:** Enables 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:** Disables 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:** Checks whether the specified DMA Channel interrupt has occurred 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).

**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\_Handle\_index***

- **TIM\_DMA\_ID\_UPDATE**  
Index of the DMA handle used for Update DMA requests
- **TIM\_DMA\_ID\_CC1**  
Index of the DMA handle used for Capture/Compare 1 DMA requests
- **TIM\_DMA\_ID\_CC2**  
Index of the DMA handle used for Capture/Compare 2 DMA requests
- **TIM\_DMA\_ID\_CC3**  
Index of the DMA handle used for Capture/Compare 3 DMA requests
- **TIM\_DMA\_ID\_CC4**  
Index of the DMA handle used for Capture/Compare 4 DMA requests
- **TIM\_DMA\_ID\_TRIGGER**  
Index of the DMA handle used for Trigger DMA requests

#### ***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
- **IS\_DMA\_MEMORY\_DATA\_SIZE**

***DMA\_Memory\_incremented\_mode***

- **DMA\_MINC\_ENABLE**  
Memory increment mode Enable
- **DMA\_MINC\_DISABLE**  
Memory increment mode Disable
- **IS\_DMA\_MEMORY\_INC\_STATE**

***DMA\_mode***

- **DMA\_NORMAL**  
Normal Mode
- **DMA\_CIRCULAR**  
Circular Mode
- **IS\_DMA\_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
- **IS\_DMA\_PERIPHERAL\_DATA\_SIZE**

***DMA\_Peripheral\_incremented\_mode***

- **DMA\_PINC\_ENABLE**  
Peripheral increment mode Enable
- **DMA\_PINC\_DISABLE**  
Peripheral increment mode Disable
- **IS\_DMA\_PERIPHERAL\_INC\_STATE**

***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
- **IS\_DMA\_PRIORITY**

***DMA Private Constants***

- **HAL\_TIMEOUT\_DMA\_ABORT**

## 15 HAL DMA Extension Driver

### 15.1 DMAEx Firmware driver defines

#### 15.1.1 DMAEx

DMAEx

##### *DMAEx Exported Macros*

- **`__HAL_DMA_GET_TC_FLAG_INDEX`**  
**Description:** Returns the current DMA Channel transfer complete flag.  
**Parameters:** `__HANDLE__`: DMA handle  
**Return value:** The specified transfer complete flag index.
- **`__HAL_DMA_GET_HT_FLAG_INDEX`**  
**Description:** Returns the current DMA Channel half transfer complete flag.  
**Parameters:** `__HANDLE__`: DMA handle  
**Return value:** The specified half transfer complete flag index.
- **`__HAL_DMA_GET_TE_FLAG_INDEX`**  
**Description:** Returns the current DMA Channel transfer error flag.  
**Parameters:** `__HANDLE__`: DMA handle  
**Return value:** The specified transfer error flag index.
- **`__HAL_DMA_GET_FLAG`**  
**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 Where x can be 1\_7 or 1\_5 to select the DMA Channel flag.  
**Return value:** The state of FLAG (SET or RESET).
- **`__HAL_DMA_CLEAR_FLAG`**  
**Description:** Clears 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 Where x can be 1\_7 or 1\_5 to select the DMA Channel flag.  
**Return value:** None:

## 16 HAL FLASH Generic Driver

### 16.1 FLASH Firmware driver registers structures

#### 16.1.1 FLASH\_ProcTypeDef

*FLASH\_ProcTypeDef* is defined in the `stm32l1xx_hal_flash.h`

##### Data Fields

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

##### Field Documentation

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

### 16.2 FLASH Firmware driver API description

The following section lists the various functions of the FLASH library.

#### 16.2.1 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

#### 16.2.2 How to use this driver

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

1. FLASH Memory 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:
  - Lock and Unlock the Flash Option bytes.
  - Set/Reset the write protection.
  - Set the Read protection Level.
  - Set the BOR level.
  - Program the user option Bytes.
  - Launch the Option Bytes loader.
  - Get the Write protection.
  - Get the read protection status.
  - Get the BOR level.
  - Get the user option bytes.
4. Interrupts and flags management functions :
  - Handle FLASH interrupts by calling HAL\_FLASH\_IRQHandler()
  - Wait for last FLASH operation according to its status
  - Get error flag status by calling HAL\_GetErrorCode()
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

### 16.2.3 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).

#### 16.2.4 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). \*

#### 16.2.5 Peripheral Control functions

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

- `HAL_FLASH_Unlock()`
- `HAL_FLASH_Lock()`
- `HAL_FLASH_OB_Unlock()`
- `HAL_FLASH_OB_Lock()`
- `HAL_FLASH_OB_Launch()`

#### 16.2.6 Peripheral Errors functions

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

- `HAL_FLASH_GetError()`

## 16.2.7 HAL\_FLASH\_Program

|                      |                                                                                                                                                                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASH_Program ( uint32_t TypeProgram, uint32_t Address, uint64_t Data)</b>                                                                                                                                                                                                               |
| Function Description | Program word at a specified address.                                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TypeProgram</b> : Indicate the way to program at a specified address. This parameter can be a value of FLASH Type Program</li> <li>• <b>Address</b> : specifies the address to be programmed.</li> <li>• <b>Data</b> : specifies the data to be programmed</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL_StatusTypeDef HAL_Status</b></li> </ul>                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• 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).</li> </ul>                            |

## 16.2.8 HAL\_FLASH\_Program\_IT

|                      |                                                                                                                                                                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASH_Program_IT ( uint32_t TypeProgram, uint32_t Address, uint64_t Data)</b>                                                                                                                                                                                                            |
| Function Description | Program word at a specified address with interrupt enabled.                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TypeProgram</b> : Indicate the way to program at a specified address. This parameter can be a value of FLASH Type Program</li> <li>• <b>Address</b> : specifies the address to be programmed.</li> <li>• <b>Data</b> : specifies the data to be programmed</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL_StatusTypeDef HAL_Status</b></li> </ul>                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                         |

## 16.2.9 HAL\_FLASH\_EndOfOperationCallback

|               |                                                                      |
|---------------|----------------------------------------------------------------------|
| Function Name | <b>void HAL_FLASH_EndOfOperationCallback ( uint32_t ReturnValue)</b> |
|---------------|----------------------------------------------------------------------|

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | FLASH end of operation interrupt callback.                                                                                                                                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>ReturnValue</b> : The value saved in this parameter depends on the ongoing procedure           <ul style="list-style-type: none"> <li>– <b>Pages Erase</b> Sector which has been erased (if 0xFFFFFFFF, it means that all the selected sectors have been erased)</li> <li>– <b>Program</b> Address which was selected for data program</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>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                 |

### 16.2.10 HAL\_FLASH\_OperationErrorCallback

|                      |                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_FLASH_OperationErrorCallback ( uint32_t ReturnValue)</b>                                                                                                                                                                                                                                                                      |
| Function Description | FLASH operation error interrupt callback.                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>ReturnValue</b> : The value saved in this parameter depends on the ongoing procedure           <ul style="list-style-type: none"> <li>– <b>Pages Erase</b> Sector number which returned an error</li> <li>– <b>Program</b> Address which was selected for data program</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>• None.</li> </ul>                                                                                                                                                                                                                                                                                 |

### 16.2.11 HAL\_FLASH\_IRQHandler

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| Function Name        | <b>void HAL_FLASH_IRQHandler ( void )</b>                 |
| Function Description | This function handles FLASH interrupt request.            |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul> |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul> |

### 16.2.12 HAL\_FLASH\_Unlock

|                      |                                                                                       |
|----------------------|---------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASH_Unlock ( void )</b>                                    |
| Function Description | Unlock the FLASH control register access.                                             |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL_StatusTypeDef HAL_Status</b></li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                               |

### 16.2.13 HAL\_FLASH\_Lock

|                      |                                                                                       |
|----------------------|---------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASH_Lock ( void )</b>                                      |
| Function Description | Locks the FLASH control register access.                                              |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL_StatusTypeDef HAL_Status</b></li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                               |

### 16.2.14 HAL\_FLASH\_OB\_Unlock

|                      |                                                                                       |
|----------------------|---------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASH_OB_Unlock ( void )</b>                                 |
| Function Description | Unlock the FLASH Option Control Registers access.                                     |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL_StatusTypeDef HAL_Status</b></li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                               |

### 16.2.15 HAL\_FLASH\_OB\_Lock

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASH_OB_Lock ( void )</b> |
| Function Description | Lock the FLASH Option Control Registers access.     |

---

|               |                                                                                       |
|---------------|---------------------------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>• <b>HAL_StatusTypeDef HAL_Status</b></li></ul> |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                               |

### 16.2.16 HAL\_FLASH\_OB\_Launch

|                      |                                                                                       |
|----------------------|---------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASH_OB_Launch ( void )</b>                                 |
| Function Description | Launch the option byte loading.                                                       |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL_StatusTypeDef HAL_Status</b></li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                               |

### 16.2.17 HAL\_FLASH\_GetError

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>FLASH_ErrorTypeDef HAL_FLASH_GetError ( void )</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Get the specific FLASH error flag.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Return values        | <ul style="list-style-type: none"><li>• <b>FLASH_ErrorCode</b> : The returned value can be:<ul style="list-style-type: none"><li>– <b>FLASH_ERROR_WRP</b>: <i>FLASH Write protected error flag</i></li><li>– <b>FLASH_ERROR_PGA</b>: <i>FLASH Programming Alignment error flag</i></li><li>– <b>FLASH_ERROR_SIZE</b>: <i>FLASH Size error flag</i></li><li>– <b>FLASH_ERROR_OPTV</b>: <i>Option validity error flag</i></li><li>– <b>FLASH_ERROR_OPTVUSR</b>: <i>Option UserValidity Error flag (available only Cat.3, Cat.4 and Cat.5 devices)</i></li><li>– <b>FLASH_ERROR_RD</b>: <i>FLASH Read Protection error flag (PCROP) (available only Cat.2 and Cat.3 devices)</i></li></ul></li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## 16.3 FLASH Firmware driver defines

### 16.3.1 FLASH

**FLASH**

***FLASH Exported Constants***

- **HAL\_FLASH\_TIMEOUT\_VALUE**
- **FLASH\_PAGE\_SIZE**

***FLASH Flags***

- **FLASH\_FLAG\_BSY**  
FLASH Busy flag
- **FLASH\_FLAG\_EOP**  
FLASH End of Programming flag
- **FLASH\_FLAG\_ENDHV**  
FLASH End of High Voltage flag
- **FLASH\_FLAG\_READY**  
FLASH Ready flag after low power mode
- **FLASH\_FLAG\_WRPERR**  
FLASH Write protected error flag
- **FLASH\_FLAG\_PGAERR**  
FLASH Programming Alignment error flag
- **FLASH\_FLAG\_SIZERR**  
FLASH Size error flag
- **FLASH\_FLAG\_OPTVERR**  
FLASH Option Validity error flag

***FLASH Interrupts***

- **\_\_HAL\_FLASH\_ENABLE\_IT**

**Description:** 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\_WRPERR: FLASH Write protected error flag  
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 UserValidity (available only Cat.3, Cat.4 and Cat.5 devices)  
FLASH\_FLAG\_RDERR : FLASH Read Protection error flag (PCROP) (available only Cat.2 and Cat.3 devices)

**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_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_WRPERR`: FLASH Write protected error flag `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 UserValidity (available only Cat.3, Cat.4 and Cat.5 devices) `FLASH_FLAG_RDERR` : FLASH Read Protection error flag (PCROP) (available only Cat.2 and Cat.3 devices)

**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
- **`IS_FLASH_LATENCY`**

#### ***FLASH Type Program***

- **`TYPEPROGRAM_WORD`**  
Program a word (32-bit) at a specified address
- **`IS_TYPEPROGRAMFLASH`**

## 17 HAL FLASH Extension Driver

### 17.1 FLASHEx Firmware driver registers structures

#### 17.1.1 FLASH\_EraseInitTypeDef

*FLASH\_EraseInitTypeDef* is defined in the `stm32l1xx_hal_flash_ex.h`

##### 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)

#### 17.1.2 FLASH\_OBProgramInitTypeDef

*FLASH\_OBProgramInitTypeDef* is defined in the `stm32l1xx_hal_flash_ex.h`

##### 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](#)

### 17.1.3 **FLASH\_AdvOBProgramInitTypeDef**

*FLASH\_AdvOBProgramInitTypeDef* is defined in the `stm32l1xx_hal_flash_ex.h`

#### 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](#)

## 17.2 **FLASHEx Firmware driver API description**

The following section lists the various functions of the FLASHEx library.

### 17.2.1 **Flash peripheral Extended features**

Comparing to other products, the FLASH interface for STM32L1xx devices contains the following additional features

- Erase functions
- DATA EEPROM memory management
- BOOT option bit configuration
- PCROP protection for all sectors

### 17.2.2 **How to use this driver**

This driver provides functions to configure and program the FLASH memory of all STM32L1xx. It includes:

- Full DATA\_EEPROM erase and program management
- Boot activation
- PCROP protection configuration and control for all pages

### 17.2.3 FLASH Erasing Programming functions

The FLASH Memory Erasing functions, includes the following functions:

- `HAL_FLASHEx_Erase`: return only when erase has been done
- `HAL_FLASHEx_Erase_IT`: end of erase is done when `HAL_FLASH_EndOfOperationCallback` is called with parameter 0xFFFFFFFF

Any operation of erase 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.
3. Call the `HAL_FLASH_Lock()` to disable the flash program memory access (recommended to protect the FLASH memory against possible unwanted operation).
- `HAL_FLASHEx_Erase()`
- `HAL_FLASHEx_Erase_IT()`

### 17.2.4 Option Bytes Programming functions

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 following function to program the desired option bytes.
  - `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 `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 (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:
  - `HAL_FLASHEx_AdvOBProgram` in selecting sectors to be read/write protected
  - `HAL_FLASHEx_OB_SelectPCROP` to enable the read/write protection
5. PcROP is available only in STM32L151xBA, STM32L152xBA, STM32L151xC, STM32L152xC & STM32L162xC devices.
  - `HAL_FLASHEx_OBProgram()`
  - `HAL_FLASHEx_OBGetConfig()`
  - `HAL_FLASHEx_AdvOBProgram()`

- [\*\*\*HAL\\_FLASHEx\\_AdvOBGetConfig\(\)\*\*\*](#)

### 17.2.5 DATA EEPROM Programming functions

Any operation of erase or program should follow these steps:

1. Call the 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 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).
- [\*\*\*HAL\\_FLASHEx\\_DATAEEPROM\\_Unlock\(\)\*\*\*](#)
  - [\*\*\*HAL\\_FLASHEx\\_DATAEEPROM\\_Lock\(\)\*\*\*](#)
  - [\*\*\*HAL\\_FLASHEx\\_DATAEEPROM\\_Erase\(\)\*\*\*](#)
  - [\*\*\*HAL\\_FLASHEx\\_DATAEEPROM\\_Program\(\)\*\*\*](#)
  - [\*\*\*HAL\\_FLASHEx\\_DATAEEPROM\\_EnableFixedTimeProgram\(\)\*\*\*](#)
  - [\*\*\*HAL\\_FLASHEx\\_DATAEEPROM\\_DisableFixedTimeProgram\(\)\*\*\*](#)

### 17.2.6 HAL\_FLASHEx\_Erase

|                      |                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASHEx_Erase (</b><br><b><i>FLASH_EraseInitTypeDef</i> * pEraseInit, uint32_t * PageError)</b>                                                                                                                                                                                                                                                                      |
| Function Description | Erase the specified FLASH memory Pages.                                                                                                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>pEraseInit</b> : pointer to an <i>FLASH_EraseInitTypeDef</i> structure that contains the configuration information for the erasing.</li> <li>• <b>PageError</b> : pointer to variable that contains the configuration information on faulty sector in case of error (0xFFFFFFFF means that all the sectors have been correctly erased)</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL_StatusTypeDef HAL_Status</b></li> </ul>                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• 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)</li> </ul>                                                                                                         |

### 17.2.7 HAL\_FLASHEx\_Erase\_IT

|               |                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_FLASHEx_Erase_IT (</b><br><b><i>FLASH_EraseInitTypeDef</i> * pEraseInit)</b> |
|---------------|-------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Perform a page erase of the specified FLASH memory pages with interrupt enabled.                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li><b>pEraseInit</b> : pointer to an <code>FLASH_EraseInitTypeDef</code> structure that contains the configuration information for the erasing.</li> </ul>                                                                                                                |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL_StatusTypeDef HAL Status</b></li> </ul>                                                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>To correctly run this function, the <code>HAL_FLASH_Unlock()</code> function must be called before. Call the <code>HAL_FLASH_Lock()</code> to disable the flash memory access (recommended to protect the FLASH memory against possible unwanted operation)</li> </ul> |

## 17.2.8 HAL\_FLASHEx\_OBProgram

|                      |                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASHEx_OBProgram (</b><br><b><i>FLASH_OBProgramInitTypeDef * pOBInit</i></b> )                                                                                    |
| Function Description | Program option bytes.                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li><b>pOBInit</b> : pointer to an <code>FLASH_OBInitStruct</code> structure that contains the configuration information for the programming.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL_StatusTypeDef HAL Status</b></li> </ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                     |

## 17.2.9 HAL\_FLASHEx\_OBGetConfig

|                      |                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_FLASHEx_OBGetConfig (</b><br><b><i>FLASH_OBProgramInitTypeDef * pOBInit</i></b> )                                                                                               |
| Function Description | Get the Option byte configuration.                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li><b>pOBInit</b> : pointer to an <code>FLASH_OBInitStruct</code> structure that contains the configuration information for the programming.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                     |

### 17.2.10 HAL\_FLASHEx\_AdvOBProgram

|                      |                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASHEx_AdvOBProgram (</b><br><i>FLASH_AdvOBProgramInitTypeDef * pAdvOBInit)</i>                                                                                             |
| Function Description | Program option bytes.                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>pAdvOBInit</b> : pointer to an <i>FLASH_AdvOBProgramInitTypeDef</i> structure that contains the configuration information for the programming.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL_StatusTypeDef HAL_Status</b></li> </ul>                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• This function can be used only for Cat2 &amp; Cat3 devices for PCROP and Cat4 &amp; Cat5 for BFB2.</li> </ul>                                                |

### 17.2.11 HAL\_FLASHEx\_AdvOBGetConfig

|                      |                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_FLASHEx_AdvOBGetConfig (</b><br><i>FLASH_AdvOBProgramInitTypeDef * pAdvOBInit)</i>                                                                                                        |
| Function Description | Get the OBEX byte configuration.                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>pAdvOBInit</b> : pointer to an <i>FLASH_AdvOBProgramInitTypeDef</i> structure that contains the configuration information for the programming.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• This function can be used only for Cat2 &amp; Cat3 devices for PCROP and Cat4 &amp; Cat5 for BFB2.</li> </ul>                                                |

### 17.2.12 HAL\_FLASHEx\_DATAEEPROM\_Unlock

|                      |                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASHEx_DATAEEPROM_Unlock (</b><br><i>void )</i>               |
| Function Description | Unlocks the data memory and <i>FLASH_PECR</i> register access.                          |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL_StatusTypeDef HAL_Status</b></li> </ul> |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                               |

### 17.2.13 HAL\_FLASHEx\_DATAEEPROM\_Lock

|                      |                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASHEx_DATAEEPROM_Lock ( void )</b>                           |
| Function Description | Locks the Data memory and FLASH_PECR register access.                                   |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL_StatusTypeDef HAL_Status</b></li> </ul> |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                               |

### 17.2.14 HAL\_FLASHEx\_DATAEEPROM\_Erase

|                      |                                                                                                                                                                                                                                                                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_FLASHEx_DATAEEPROM_Erase ( uint32_t TypeErase, uint32_t Address)</b>                                                                                                                                                                                                                                   |
| Function Description | Erase a word in data memory.                                                                                                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Address</b> : specifies the address to be erased.</li> <li>• <b>TypeErase</b> : Indicate the way to erase at a specified address. This parameter can be a value of FLASH Type Program</li> </ul>                                                                                    |
| Return values        | <ul style="list-style-type: none"> <li>• <b>FLASH Status:</b> The returned value can be:<br/><b>FLASH_ERROR_PROGRAM, FLASH_ERROR_WRP, FLASH_COMPLETE or FLASH_TIMEOUT.</b></li> </ul>                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• To correctly run this function, the DATA EEPROM Unlock() function must be called before. Call the DATA EEPROM Lock() to the data EEPROM access and Flash program erase control register access(recommended to protect the DATA EEPROM against possible unwanted operation).</li> </ul> |

### 17.2.15 HAL\_FLASHEx\_DATAEEPROM\_Program

|               |                                                         |
|---------------|---------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_FLASHEx_DATAEEPROM_Program</b> |
|---------------|---------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <b>( uint32_t TypeProgram, uint32_t Address, uint32_t Data )</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Function Description | Program word at a specified address.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TypeProgram</b> : Indicate the way to program at a specified address. This parameter can be a value of FLASHEx Type Program Data</li> <li>• <b>Address</b> : specifies the address to be programmed.</li> <li>• <b>Data</b> : specifies the data to be programmed</li> </ul>                                                                                                                                                                                                                    |
| Return values        | <b>HAL_StatusTypeDef HAL_Status</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>• To correctly run this function, the <code>HAL_FLASH_EEPROM_Unlock()</code> function must be called before. Call the <code>HAL_FLASHEx_DATAEEPROM_Unlock()</code> to he data EEPROM access and Flash program erase control register access(recommended to protect the DATA_EEPROM against possible unwanted operation).</li> <li>• The function <code>HAL_FLASHEx_DATAEEPROM_EnableFixedTimeProgram()</code> can be called before this function to configure the Fixed Time Programming.</li> </ul> |

### 17.2.16 HAL\_FLASHEx\_DATAEEPROM\_EnableFixedTimeProgram

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| Function Name        | <b>void<br/>HAL_FLASHEx_DATAEEPROM_EnableFixedTimeProgram ( void )</b> |
| Function Description | Enable DATA EEPROM fixed Time programming (2*Tprog).                   |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>              |

### 17.2.17 HAL\_FLASHEx\_DATAEEPROM\_DisableFixedTimeProgram

|                      |                                                                         |
|----------------------|-------------------------------------------------------------------------|
| Function Name        | <b>void<br/>HAL_FLASHEx_DATAEEPROM_DisableFixedTimeProgram ( void )</b> |
| Function Description | Disables DATA EEPROM fixed Time programming (2*Tprog).                  |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>               |

## 17.3 FLASHEx Firmware driver defines

### 17.3.1 FLASHEx

FLASHEx

*FLASHEx Address*

- **FLASH\_NBPGES\_MAX**
- **IS\_FLASH\_DATA\_ADDRESS**
- **IS\_FLASH\_PROGRAM\_ADDRESS**
- **IS\_FLASH\_PROGRAM\_BANK1\_ADDRESS**
- **IS\_FLASH\_PROGRAM\_BANK2\_ADDRESS**
- **IS\_NBPGES**

*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**  
**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**  
**Return value:**none
- **\_HAL\_FLASH\_ACC64\_DISABLE**  
**Return value:**none
- **\_HAL\_FLASH\_PREFETCH\_BUFFER\_ENABLE**  
**Return value:**none
- **\_HAL\_FLASH\_PREFETCH\_BUFFER\_DISABLE**  
**Return value:**none
- **\_HAL\_FLASH\_SLEEP\_POWERDOWN\_ENABLE**  
**Return value:**none
- **\_HAL\_FLASH\_SLEEP\_POWERDOWN\_DISABLE**  
**Return value:**none
- **\_HAL\_FLASH\_POWER\_DOWN\_ENABLE**
- **\_HAL\_FLASH\_POWER\_DOWN\_DISABLE**

*FLASHEx Flags*

- **FLASH\_FLAG\_OPTVERRUSR**  
FLASH Option User Validity error flag
- **FLASH\_FLAG\_MASK**

*FLASHEx Option Advanced Type*

- **OBEX\_BOOTCONFIG**  
BOOTConfig option byte configuration

- **IS\_OBEX**

#### ***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)

- **IS\_OB\_BOOT\_BANK**

#### ***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

- **IS\_OB\_BOR\_LEVEL**

#### ***FLASHEx Option Bytes IWatchdog***

- **OB\_IWDG\_SW**

Software WDG selected

- **OB\_IWDG\_HW**

Hardware WDG selected

- **IS\_OB\_IWDG\_SOURCE**

#### ***FLASHEx Option Bytes nRST\_STDBY***

- **OB\_STDBY\_NORST**

No reset generated when entering in STANDBY

- **OB\_STDBY\_RST**

Reset generated when entering in STANDBY

- **IS\_OB\_STDBY\_SOURCE**

#### ***FLASHEx Option Bytes nRST\_STOP***

- **OB\_STOP\_NORST**

No reset generated when entering in STOP

- **OB\_STOP\_RST**

Reset generated when entering in STOP

- **IS\_OB\_STOP\_SOURCE**

#### ***FLASHEx Option Bytes Read Protection***

- **OB\_RDP\_LEVEL0**

- **OB\_RDP\_LEVEL1**

- **IS\_OB\_RDP**

***FLASHEx Option Bytes Write Mask***

- **WRP\_MASK\_LOW**
- **WRP\_MASK\_HIGH**

***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

Write protection of all Sectors

#### *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
- **IS\_OPTIONBYTE**

#### *FLASHEx\_Type\_Erase*

- **TYPEERASE\_PAGES**  
Page erase only
- **IS\_TYPEERASE**

#### *FLASHEx\_Type\_Erase Data*

- **TYPEERASEDATA\_BYTE**  
Erase byte (8-bit) at a specified address.
- **TYPEERASEDATA\_HALFWORD**  
Erase a half-word (16-bit) at a specified address.
- **TYPEERASEDATA\_WORD**  
Erase a word (32-bit) at a specified address.
- **IS\_TYPEERASEDATA**

#### *FLASHEx\_Type\_Program Data*

- **TYPEPROGRAMDATA\_BYTE**  
Program byte (8-bit) at a specified address.
- **TYPEPROGRAMDATA\_HALFWORD**  
Program a half-word (16-bit) at a specified address.
- **TYPEPROGRAMDATA\_WORD**  
Program a word (32-bit) at a specified address.
- **TYPEPROGRAMDATA\_FASTBYTE**  
Fast Program byte (8-bit) at a specified address.
- **TYPEPROGRAMDATA\_FASTHALFWORD**  
Fast Program a half-word (16-bit) at a specified address.
- **TYPEPROGRAMDATA\_FASTWORD**  
Fast Program a word (32-bit) at a specified address.
- **IS\_TYPEPROGRAMDATA**

#### *FLASHEx\_WRP State*

- **WRPSTATE\_DISABLE**  
Disable the write protection of the desired bank 1 sectors
- **WRPSTATE\_ENABLE**  
Enable the write protection of the desired bank 1 sectors
- **IS\_WRPSTATE**

## 18 HAL FLASH\_\_RAMFUNC Generic Driver

### 18.1 FLASH\_\_RAMFUNC Firmware driver API description

The following section lists the various functions of the FLASH\_\_RAMFUNC library.

#### 18.1.1 HAL\_FLASHEx\_EnableRunPowerDown

|                      |                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>__RAM_FUNC HAL_FLASHEx_EnableRunPowerDown ( void )</code>                                                                  |
| Function Description | Enable the power down mode during RUN mode.                                                                                      |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                            |
| Notes                | <ul style="list-style-type: none"><li>This function can be used only when the user code is running from Internal SRAM.</li></ul> |

#### 18.1.2 HAL\_FLASHEx\_DisableRunPowerDown

|                      |                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>__RAM_FUNC HAL_FLASHEx_DisableRunPowerDown ( void )</code>                                                                 |
| Function Description | Disable the power down mode during RUN mode.                                                                                     |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                            |
| Notes                | <ul style="list-style-type: none"><li>This function can be used only when the user code is running from Internal SRAM.</li></ul> |

#### 18.1.3 HAL\_FLASHEx\_EraseParallelPage

|                      |                                                                                                                                                                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>__RAM_FUNC HAL_FLASHEx_EraseParallelPage ( uint32_t Page_Address1, uint32_t Page_Address2)</code>                                                                                                                                                                                                                                      |
| Function Description | Erases a specified 2 page in program memory in parallel.                                                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li><b>Page_Address1</b> : 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.</li><li><b>Page_Address2</b> : The page address in program memory to be erased in the second Bank (BANK2). This parameter</li></ul> |

---

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | should be between FLASH_BANK2_BASE and FLASH_BANK2_END.                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Return values | <ul style="list-style-type: none"> <li><b>HAL Status: The returned value can be: HAL_ERROR, HAL_OK or HAL_TIMEOUT.</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                   |
| Notes         | <ul style="list-style-type: none"> <li>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).</li> <li>A Page is erased in the Program memory only if the address to load is the start address of a page (multiple of 256 bytes).</li> </ul> |

### 18.1.4 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 | Programs 2 half page in program memory in parallel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li><b>Address1</b> : 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).</li> <li><b>pBuffer1</b> : pointer to the buffer containing the data to be written to the first half page in the first bank.</li> <li><b>Address2</b> : 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).</li> <li><b>pBuffer2</b> : pointer to the buffer containing the data to be written to the second half page in the second bank.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL Status: The returned value can be: HAL_ERROR, HAL_OK or HAL_TIMEOUT.</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>This function can be used only for STM32L151xD, STM32L152xD, STM32L162xD and Cat5 devices.</li> <li>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).</li> <li>Half page write is possible only from SRAM.</li> <li>If there are more than 32 words to write, after 32 words another Half Page programming operation starts and has to be finished.</li> <li>A half page is written to the program memory only if the first</li> </ul>                         |

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 PGAERR 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.

### 18.1.5 HAL\_FLASHEx\_HalfPageProgram

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>__RAM_FUNC HAL_FLASHEx_HalfPageProgram ( uint32_t Address, uint32_t * pBuffer)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Function Description | Programs a half page in program memory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Address</b> : specifies the address to be written.</li> <li>• <b>pBuffer</b> : pointer to the buffer containing the data to be written to the half page.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL Status:</b> The returned value can be: <b>HAL_ERROR</b>, <b>HAL_OK</b> or <b>HAL_TIMEOUT</b>.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• To correctly run this function, the <code>HAL_FLASH_Unlock()</code> function must be called before. Call the <code>HAL_FLASH_Lock()</code> to disable the flash memory access (recommended to protect the FLASH memory against possible unwanted operation)</li> <li>• Half page write is possible only from SRAM.</li> <li>• If there are more than 32 words to write, after 32 words another Half Page programming operation starts and has to be finished.</li> <li>• 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.</li> <li>• 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.).</li> <li>• If a PGAERR 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.</li> </ul> |

### 18.1.6 HAL\_FLASHEx\_DATAEEPROM\_EraseDoubleWord

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>__RAM_FUNC<br/>HAL_FLASHEx_DATAEEPROM_EraseDoubleWord ( uint32_t Address)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Function Description | Erase a double word in data memory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li><b>Address</b> : specifies the address to be erased.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL Status:</b> The returned value can be: <code>HAL_ERROR</code>, <code>HAL_OK</code> or <code>HAL_TIMEOUT</code>.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>To correctly run this function, the <code>HAL_FLASH_EEPROM_Unlock()</code> function must be called before. Call the <code>HAL_FLASH_EEPROM_Lock()</code> to he data EEPROM access and Flash program erase control register access(recommended to protect the DATA_EEPROM against possible unwanted operation).</li> <li>Data memory double word erase is possible only from SRAM.</li> <li>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).</li> <li>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.).</li> </ul> |

### 18.1.7 HAL\_FLASHEx\_DATAEEPROM\_ProgramDoubleWord

|                      |                                                                                                                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>__RAM_FUNC<br/>HAL_FLASHEx_DATAEEPROM_ProgramDoubleWord ( uint32_t Address, uint64_t Data)</code>                                                                                                                                                                                                                            |
| Function Description | Write a double word in data memory without erase.                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li><b>Address</b> : specifies the address to be written.</li> <li><b>Data</b> : specifies the data to be written.</li> </ul>                                                                                                                                                                   |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL Status:</b> The returned value can be: <code>HAL_ERROR</code>, <code>HAL_OK</code> or <code>HAL_TIMEOUT</code>.</li> </ul>                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>To correctly run this function, the <code>HAL_FLASH_EEPROM_Unlock()</code> function must be called before. Call the <code>HAL_FLASH_EEPROM_Lock()</code> to he data EEPROM access and Flash program erase control register access(recommended to protect the DATA_EEPROM against</li> </ul> |

- 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.).

## 19 HAL GPIO Generic Driver

### 19.1 GPIO Firmware driver registers structures

#### 19.1.1 GPIO\_InitTypeDef

*GPIO\_InitTypeDef* is defined in the `stm32l1xx_hal_gpio.h`

##### 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\\_define\*](#)
- *uint32\_t GPIO\_InitTypeDef::Mode* Specifies the operating mode for the selected pins. This parameter can be a value of [\*GPIO\\_mode\\_define\*](#)
- *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\\_define\*](#)
- *uint32\_t GPIO\_InitTypeDef::Speed* Specifies the speed for the selected pins. This parameter can be a value of [\*GPIO\\_speed\\_define\*](#)
- *uint32\_t GPIO\_InitTypeDef::Alternate* Peripheral to be connected to the selected pins This parameter can be a value of [\*GPIOEx\\_Alternate\\_function\\_selection\*](#)

### 19.2 GPIO Firmware driver API description

The following section lists the various functions of the GPIO library.

#### 19.2.1 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 23 edge detectors (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.

## 19.2.2 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
  - 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. To get the level of a pin configured in input mode use HAL\_GPIO\_ReadPin().
5. To set/reset the level of a pin configured in output mode use HAL\_GPIO\_WritePin()/HAL\_GPIO\_TogglePin().
6. To lock pin configuration until next reset use HAL\_GPIO\_LockPin().
7. During and just after reset, the alternate functions are not active and the GPIO pins are configured in input floating mode (except JTAG pins).
8. 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.
9. 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.

## 19.2.3 Initialization and de-initialization functions

- [\*\*HAL\\_GPIO\\_Init\(\)\*\*](#)
- [\*\*HAL\\_GPIO\\_DeInit\(\)\*\*](#)

## 19.2.4 HAL\_GPIO\_Init

|                      |                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_GPIO_Init ( GPIO_TypeDef * GPIOx,<br/>GPIO_InitTypeDef * GPIO_InitStruct)</b>                                                                                                                                                                                                                                               |
| Function Description | Initializes the GPIOx peripheral according to the specified parameters in the GPIO_Init.                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>GPIOx</b> : where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices</li> <li>• <b>GPIO_InitStruct</b> : pointer to a GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                               |

## 19.2.5 HAL\_GPIO\_DeInit

|                      |                                                                                                                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_GPIO_DeInit ( GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)</b>                                                                                                                                                                                                                                          |
| Function Description | De-initializes the GPIOx peripheral registers to their default reset values.                                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>GPIOx</b> : where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices</li> <li>• <b>GPIO_Pin</b> : specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15).</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                       |

## 19.2.6 HAL\_GPIO\_ReadPin

|                      |                                                                                      |
|----------------------|--------------------------------------------------------------------------------------|
| Function Name        | <b>GPIO_PinState HAL_GPIO_ReadPin ( GPIO_TypeDef * GPIOx,<br/>uint16_t GPIO_Pin)</b> |
| Function Description | Reads the specified input port pin.                                                  |

|               |                                                                                                                                                                                                                                                                                                    |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>GPIOx</b> : where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices</li> <li>• <b>GPIO_Pin</b> : specifies the port bit to read. This parameter can be GPIO_PIN_x where x can be (0..15).</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• <b>The input port pin value.</b></li> </ul>                                                                                                                                                                                                               |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                          |

### 19.2.7 HAL\_GPIO\_WritePin

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_GPIO_WritePin ( GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Function Description | Sets or clears the selected data port bit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>GPIOx</b> : where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices</li> <li>• <b>GPIO_Pin</b> : specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15).</li> <li>• <b>PinState</b> : specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum values: <ul style="list-style-type: none"> <li>– <b>GPIO_BIT_RESET</b> to clear the port pin</li> <li>– <b>GPIO_BIT_SET</b> to set the port pin</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                          |

### 19.2.8 HAL\_GPIO\_TogglePin

|                      |                                                                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_GPIO_TogglePin ( GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)</b>                                                                                                                                                                   |
| Function Description | Toggles the specified GPIO pin.                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>GPIOx</b> : where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices</li> <li>• <b>GPIO_Pin</b> : Specifies the pins to be toggled.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                   |

## Notes

- None.

### 19.2.9 HAL\_GPIO\_LockPin

|                      |                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_GPIO_LockPin ( GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)</b>                                                                                                                                                                                                                                    |
| Function Description | Locks GPIO Pins configuration registers.                                                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>GPIOx</b> : where x can be (A..G depending on device used) to select the GPIO peripheral for STM32L1XX family devices</li><li>• <b>GPIO_Pin</b> : specifies the port bit to be locked. This parameter can be any combination of GPIO_Pin_x where x can be (0..15).</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• The locked registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.</li><li>• The configuration of the locked GPIO pins can no longer be modified until the next reset.</li></ul>                                                        |

### 19.2.10 HAL\_GPIO\_EXTI\_IRQHandler

|                      |                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_GPIO_EXTI_IRQHandler ( uint16_t GPIO_Pin)</b>                                                  |
| Function Description | This function handles EXTI interrupt request.                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>GPIO_Pin</b> : Specifies the pins connected EXTI line</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                    |

### 19.2.11 HAL\_GPIO\_EXTI\_Callback

|                      |                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_GPIO_EXTI_Callback ( uint16_t GPIO_Pin)</b>                                                    |
| Function Description | EXTI line detection callback.                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li><b>GPIO_Pin</b> : Specifies the pins connected EXTI line</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                    |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                    |

## 19.3 GPIO Firmware driver defines

### 19.3.1 GPIO

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 define*

- **GPIO\_MODE\_INPUT**

Input Floating Mode

- **GPIO\_MODE\_OUTPUT\_PP**

Output Push Pull Mode

- **GPIO\_MODE\_OUTPUT\_OD**

Output Open Drain Mode

- **GPIO\_MODE\_AF\_PP**

Alternate Function Push Pull Mode

- **GPIO\_MODE\_AF\_OD**  
Alternate Function Open Drain Mode
- **GPIO\_MODE\_ANALOG**  
Analog Mode
- **GPIO\_MODE\_IT\_RISING**  
External Interrupt Mode with Rising edge trigger detection
- **GPIO\_MODE\_IT\_FALLING**  
External Interrupt Mode with Falling edge trigger detection
- **GPIO\_MODE\_IT\_RISING\_FALLING**  
External Interrupt Mode with Rising/Falling edge trigger detection
- **GPIO\_MODE\_EVT\_RISING**  
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 define*

- **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 Private Constants*

- **GPIO\_MODE**
- **EXTI\_MODE**
- **GPIO\_MODE\_IT**
- **GPIO\_MODE\_EVT**
- **RISING\_EDGE**
- **FALLING\_EDGE**
- **GPIO\_OUTPUT\_TYPE**
- **GPIO\_NUMBER**

#### *GPIO\_Private\_Macros*

- **IS\_GPIO\_PIN\_ACTION**
- **IS\_GPIO\_PIN**
- **IS\_GPIO\_PULL**
- **IS\_GPIO\_SPEED**
- **IS\_GPIO\_MODE**

***GPIO pull define***

- **GPIO\_NOPULL**  
No Pull-up or Pull-down activation
- **GPIO\_PULLUP**  
Pull-up activation
- **GPIO\_PULLDOWN**  
Pull-down activation

***GPIO speed define***

- **GPIO\_SPEED\_VERY\_LOW**  
Very Low speed
- **GPIO\_SPEED\_LOW**  
Low speed
- **GPIO\_SPEED\_MEDIUM**  
Medium speed
- **GPIO\_SPEED\_HIGH**  
High speed

## 20 HAL GPIO Extension Driver

### 20.1 GPIOEx Firmware driver defines

#### 20.1.1 GPIOEx

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

***GPIOEx Private Macros***

- **IS\_GPIO\_AF**
- **GET\_GPIO\_INDEX**

## 21 HAL I2C Generic Driver

### 21.1 I2C Firmware driver registers structures

#### 21.1.1 I2C\_InitTypeDef

*I2C\_InitTypeDef* is defined in the `stm32l1xx_hal_i2c.h`

##### 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`](#)

#### 21.1.2 I2C\_HandleTypeDef

*I2C\_HandleTypeDef* is defined in the `stm32l1xx_hal_i2c.h`

##### Data Fields

- `I2C_TypeDef * Instance`
- `I2C_InitTypeDef Init`
- `uint8_t * pBuffPtr`
- `uint16_t XferSize`
- `__IO uint16_t XferCount`
- `DMA_HandleTypeDef * hdmatx`
- `DMA_HandleTypeDef * hdmarx`
- `HAL_LockTypeDef Lock`

- `__IO HAL_I2C_StateTypeDef State`
- `__IO HAL_I2C_ErrorTypeDef ErrorCode`

#### Field Documentation

- `I2C_TypeDef* 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
- `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_ErrorTypeDef I2C_HandleTypeDef::ErrorCode`

## 21.2 I2C Firmware driver API description

The following section lists the various functions of the I2C library.

### 21.2.1 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 implement 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

### Interrupt mode IO MEM operation

- Write an amount of data in no-blocking mode with Interrupt to a specific memory address using HAL\_I2C\_Mem\_Write\_IT()
- At MEM 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 no-blocking mode with Interrupt from a specific memory address using HAL\_I2C\_Mem\_Read\_IT()
- At MEM 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

### DMA mode IO MEM operation

- Write an amount of data in no-blocking mode with DMA to a specific memory address using HAL\_I2C\_Mem\_Write\_DMA()
- At MEM 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 no-blocking mode with DMA from a specific memory address using HAL\_I2C\_Mem\_Read\_DMA()
- At MEM 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_GET_FLAG` : Checks 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

## 21.2.2 Initialization and de-initialization functions

This subsection provides a set of functions allowing to initialize and de-initialize 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.
- `HAL_I2C_Init()`
- `HAL_I2C_DeInit()`
- `HAL_I2C_MspInit()`
- `HAL_I2C_MspDeInit()`

## 21.2.3 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\_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\_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\_Slave\_Transmit\_IT()*
  - *HAL\_I2C\_Slave\_Receive\_IT()*
  - *HAL\_I2C\_Master\_Transmit\_DMA()*
  - *HAL\_I2C\_Master\_Receive\_DMA()*
  - *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\_MemTxCpltCallback()*
  - *HAL\_I2C\_MemRxCpltCallback()*
  - *HAL\_I2C\_ErrorCallback()*

## 21.2.4 Peripheral State and Errors functions

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

- [\*HAL\\_I2C\\_GetState\(\)\*](#)
- [\*HAL\\_I2C\\_GetError\(\)\*](#)

## 21.2.5 HAL\_I2C\_Init

|                      |                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_Init ( <i>I2C_HandleTypeDef</i> * <i>hi2c</i>)</b>                                                                                              |
| Function Description | Initializes the I2C according to the specified parameters in the <i>I2C_InitTypeDef</i> and create the associated handle.                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for I2C module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                    |

## 21.2.6 HAL\_I2C\_DelInit

|                      |                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_DelInit ( <i>I2C_HandleTypeDef</i> * <i>hi2c</i>)</b>                                                                                           |
| Function Description | Deinitializes the I2C peripheral.                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for I2C module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                    |

## 21.2.7 HAL\_I2C\_MspInit

|                      |                                                                       |
|----------------------|-----------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_MspInit ( <i>I2C_HandleTypeDef</i> * <i>hi2c</i>)</b> |
| Function Description | I2C MSP Init.                                                         |

|               |                                                                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li></ul> |
| Return values | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 21.2.8 HAL\_I2C\_MspDeInit

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_MspDeInit ( <i>I2C_HandleTypeDef</i> * hi2c)</b>                                                                                                    |
| Function Description | I2C MSP DeInit.                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 21.2.9 HAL\_I2C\_Master\_Transmit

|                      |                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_Master_Transmit ( <i>I2C_HandleTypeDef</i> * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                                            |
| Function Description | Transmits in master mode an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li><li>• <b>DevAddress</b> : Target device address</li><li>• <b>pData</b> : Pointer to data buffer</li><li>• <b>Size</b> : Amount of data to be sent</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                               |

## 21.2.10 HAL\_I2C\_Master\_Receive

|                      |                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Master_Receive (<br/>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t *<br/>pData, uint16_t Size, uint32_t Timeout)</code>                                                                                                                                                                                                            |
| Function Description | Receives in master mode an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>DevAddress</b> : Target device address</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                   |

## 21.2.11 HAL\_I2C\_Slave\_Transmit

|                      |                                                                                                                                                                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Slave_Transmit (<br/>I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size,<br/>uint32_t Timeout)</code>                                                                                                                                                                            |
| Function Description | Transmits in slave mode an amount of data in blocking mode.                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                              |

## 21.2.12 HAL\_I2C\_Slave\_Receive

|               |                                                                                                                      |
|---------------|----------------------------------------------------------------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_I2C_Slave_Receive (<br/>I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size,</code> |
|---------------|----------------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <b>uint32_t Timeout)</b>                                                                                                                                                                                                                                                                                               |
| Function Description | Receive in slave mode an amount of data in blocking mode.                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                              |

### 21.2.13 HAL\_I2C\_Master\_Transmit\_IT

|                      |                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_Master_Transmit_IT (</b><br><b>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                                                 |
| Function Description | Transmit in master mode an amount of data in no-blocking mode with Interrupt.                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>DevAddress</b> : Target device address</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                      |

### 21.2.14 HAL\_I2C\_Master\_Receive\_IT

|                      |                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_Master_Receive_IT (</b><br><b>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)</b>                                                                              |
| Function Description | Receive in master mode an amount of data in no-blocking mode with Interrupt.                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>DevAddress</b> : Target device address</li> </ul> |

- **pData** : Pointer to data buffer
  - **Size** : Amount of data to be sent
- Return values
- **HAL status**
- Notes
- None.

### 21.2.15 HAL\_I2C\_Slave\_Transmit\_IT

|                      |                                                                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_Slave_Transmit_IT (</b><br><b>I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                  |
| Function Description | Transmit in slave mode an amount of data in no-blocking mode with Interrupt.                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                       |
| Notes                | • None.                                                                                                                                                                                                                                                                   |

### 21.2.16 HAL\_I2C\_Slave\_Receive\_IT

|                      |                                                                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_Slave_Receive_IT (</b><br><b>I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                   |
| Function Description | Receive in slave mode an amount of data in no-blocking mode with Interrupt.                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                       |
| Notes                | • None.                                                                                                                                                                                                                                                                   |

### 21.2.17 HAL\_I2C\_Master\_Transmit\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Master_Transmit_DMA (</code><br><i>I2C_HandleTypeDef</i> * <code>hi2c</code> , <code>uint16_t DevAddress</code> , <code>uint8_t * pData</code> , <code>uint16_t Size</code> )                                                                                                                                                 |
| Function Description | Transmit in master mode an amount of data in no-blocking mode with DMA.                                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <code>hi2c</code> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for I2C module</li> <li>• <code>DevAddress</code> : Target device address</li> <li>• <code>pData</code> : Pointer to data buffer</li> <li>• <code>Size</code> : Amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                     |

### 21.2.18 HAL\_I2C\_Master\_Receive\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Master_Receive_DMA (</code><br><i>I2C_HandleTypeDef</i> * <code>hi2c</code> , <code>uint16_t DevAddress</code> , <code>uint8_t * pData</code> , <code>uint16_t Size</code> )                                                                                                                                                  |
| Function Description | Receive in master mode an amount of data in no-blocking mode with DMA.                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <code>hi2c</code> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for I2C module</li> <li>• <code>DevAddress</code> : Target device address</li> <li>• <code>pData</code> : Pointer to data buffer</li> <li>• <code>Size</code> : Amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                     |

### 21.2.19 HAL\_I2C\_Slave\_Transmit\_DMA

|               |                                                             |
|---------------|-------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_I2C_Slave_Transmit_DMA (</code> |
|---------------|-------------------------------------------------------------|

***I2C\_HandleTypeDef \* hi2c, uint8\_t \* pData, uint16\_t Size)***

|                      |                                                                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Transmit in slave mode an amount of data in no-blocking mode with DMA.                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                       |
| Notes                | • None.                                                                                                                                                                                                                                                                   |

**21.2.20 HAL\_I2C\_Slave\_Receive\_DMA**

|                      |                                                                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_Slave_Receive_DMA (</b><br><b><i>I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)</i></b>                                                                                                                                           |
| Function Description | Receive in slave mode an amount of data in no-blocking mode with DMA.                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                       |
| Notes                | • None.                                                                                                                                                                                                                                                                   |

**21.2.21 HAL\_I2C\_Mem\_Write**

|                      |                                                                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2C_Mem_Write (</b><br><b><i>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size, uint32_t Timeout)</i></b>                                                                          |
| Function Description | Write an amount of data in blocking mode to a specific memory address.                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>DevAddress</b> : Target device address</li> <li>• <b>MemAddress</b> : Internal memory address</li> </ul> |

- **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**
- Notes                • None.

### 21.2.22 HAL\_I2C\_Mem\_Read

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Mem_Read (</code><br><code>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t</code><br><code>MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t</code><br><code>Size, uint32_t Timeout)</code>                                                                                                                                                                                                                                                      |
| Function Description | Read an amount of data in blocking mode from a specific memory address.                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>DevAddress</b> : Target device address</li> <li>• <b>MemAddress</b> : Internal memory address</li> <li>• <b>MemAddSize</b> : Size of internal memory address</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Notes                | • None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

### 21.2.23 HAL\_I2C\_Mem\_Write\_IT

|                      |                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Mem_Write_IT (</code><br><code>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t</code><br><code>MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t</code><br><code>Size)</code> |
| Function Description | Write an amount of data in no-blocking mode with Interrupt to a specific memory address.                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> </ul>                                                          |

- **DevAddress** : Target device address
  - **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**
- Notes
- None.

## 21.2.24 HAL\_I2C\_Mem\_Read\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Mem_Read_IT (<br/>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t<br/>MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t<br/>Size)</code>                                                                                                                                                                                                                                                            |
| Function Description | Read an amount of data in no-blocking mode with Interrupt from a specific memory address.                                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>DevAddress</b> : Target device address</li> <li>• <b>MemAddress</b> : Internal memory address</li> <li>• <b>MemAddSize</b> : Size of internal memory address</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                            |

## 21.2.25 HAL\_I2C\_Mem\_Write\_DMA

|                      |                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Mem_Write_DMA (<br/>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t<br/>MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t<br/>Size)</code> |
| Function Description | Write an amount of data in no-blocking mode with DMA to a specific memory address.                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> </ul>                       |

- **DevAddress** : Target device address
  - **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**
- Notes
- None.

### 21.2.26 HAL\_I2C\_Mem\_Read\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_Mem_Read_DMA (</code><br><code>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t</code><br><code>MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t</code><br><code>Size)</code>                                                                                                                                                                                                                       |
| Function Description | Reads an amount of data in no-blocking mode with DMA from a specific memory address.                                                                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>DevAddress</b> : Target device address</li> <li>• <b>MemAddress</b> : Internal memory address</li> <li>• <b>MemAddSize</b> : Size of internal memory address</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be read</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                            |

### 21.2.27 HAL\_I2C\_IsDeviceReady

|                      |                                                                                                                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_I2C_IsDeviceReady (</code><br><code>I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint32_t</code><br><code>Trials, uint32_t Timeout)</code>                                                                                               |
| Function Description | Checks if target device is ready for communication.                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li> <li>• <b>DevAddress</b> : Target device address</li> <li>• <b>Trials</b> : Number of trials</li> </ul> |

- **Timeout** : Timeout duration
- Return values • **HAL status**
- Notes • This function is used with Memory devices

### 21.2.28 HAL\_I2C\_EV\_IRQHandler

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_EV_IRQHandler ( <i>I2C_HandleTypeDef</i> * hi2c)</b>                                                                                                       |
| Function Description | This function handles I2C event interrupt request.                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |

### 21.2.29 HAL\_I2C\_ER\_IRQHandler

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_ER_IRQHandler ( <i>I2C_HandleTypeDef</i> * hi2c)</b>                                                                                                       |
| Function Description | This function handles I2C error interrupt request.                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |

### 21.2.30 HAL\_I2C\_MasterTxCpltCallback

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_MasterTxCpltCallback ( <i>I2C_HandleTypeDef</i> * hi2c)</b> |
| Function Description | Master Tx Transfer completed callbacks.                                     |

|               |                                                                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li></ul> |
| Return values | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 21.2.31 HAL\_I2C\_MasterRxCpltCallback

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_MasterRxCpltCallback ( I2C_HandleTypeDef * hi2c)</b>                                                                                                |
| Function Description | Master Rx Transfer completed callbacks.                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 21.2.32 HAL\_I2C\_SlaveTxCpltCallback

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_SlaveTxCpltCallback ( I2C_HandleTypeDef * hi2c)</b>                                                                                                 |
| Function Description | Slave Tx Transfer completed callbacks.                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDef structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 21.2.33 HAL\_I2C\_SlaveRxCpltCallback

|               |                                                                     |
|---------------|---------------------------------------------------------------------|
| Function Name | <b>void HAL_I2C_SlaveRxCpltCallback ( I2C_HandleTypeDef * hi2c)</b> |
|---------------|---------------------------------------------------------------------|

**hi2c)**

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Slave Rx Transfer completed callbacks.                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDefDef structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |

### 21.2.34 HAL\_I2C\_MemTxCpltCallback

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_MemTxCpltCallback ( <i>I2C_HandleTypeDefDef</i> *<br/>hi2c)</b>                                                                                        |
| Function Description | Memory Tx Transfer completed callbacks.                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDefDef structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |

### 21.2.35 HAL\_I2C\_MemRxCpltCallback

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_MemRxCpltCallback ( <i>I2C_HandleTypeDefDef</i> *<br/>hi2c)</b>                                                                                        |
| Function Description | Memory Rx Transfer completed callbacks.                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a I2C_HandleTypeDefDef structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |

### 21.2.36 HAL\_I2C\_ErrorCallback

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2C_ErrorCallback ( <i>I2C_HandleTypeDef</i> * hi2c)</b>                                                                                                       |
| Function Description | I2C error callbacks.                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |

### 21.2.37 HAL\_I2C\_GetState

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_I2C_StateTypeDef HAL_I2C_GetState ( <i>I2C_HandleTypeDef</i> * hi2c)</b>                                                                                            |
| Function Description | Returns the I2C state.                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for I2C module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                                                                                         |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |

### 21.2.38 HAL\_I2C\_GetError

|                      |                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_I2C_GetError ( <i>I2C_HandleTypeDef</i> * hi2c)</b>                                                                                                                |
| Function Description | Return the I2C error code.                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2c</b> : pointer to a <i>I2C_HandleTypeDef</i> structure that contains the configuration information for the specified I2C.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>I2C Error Code</b></li></ul>                                                                                                            |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                            |

## 21.3 I2C Firmware driver defines

### 21.3.1 I2C

I2C

*I2C\_addressing\_mode*

- I2C\_ADDRESSINGMODE\_7BIT
- I2C\_ADDRESSINGMODE\_10BIT
- IS\_I2C\_ADDRESSING\_MODE

*I2C\_Clock\_Speed\_definition*

- IS\_I2C\_CLOCK\_SPEED

*I2C\_dual\_addressing\_mode*

- I2C\_DUALADDRESS\_DISABLED
- I2C\_DUALADDRESS\_ENABLED
- IS\_I2C\_DUAL\_ADDRESS

*I2C\_duty\_cycle\_in\_fast\_mode*

- I2C\_DUTYCYCLE\_2
- I2C\_DUTYCYCLE\_16\_9
- IS\_I2C\_DUTY\_CYCLE

*I2C Exported Macros*

- \_\_HAL\_I2C\_RESET\_HANDLE\_STATE

**Description:** Reset I2C handle state.

**Parameters:** \_\_HANDLE\_\_: specifies the I2C Handle. This parameter can be I2Cx where x: 1, 2, or 3 to select the I2C peripheral.

**Return value:**None

- \_\_HAL\_I2C\_ENABLE\_IT

**Description:** Enable or disable the specified I2C interrupts.

**Parameters:** \_\_HANDLE\_\_: specifies the I2C Handle. This parameter can be I2Cx where x: 1 or 2 to select the I2C peripheral. \_\_INTERRUPT\_\_: specifies the interrupt source to enable or disable. 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:**None

- \_\_HAL\_I2C\_DISABLE\_IT

- \_\_HAL\_I2C\_GET\_IT\_SOURCE

**Description:** Checks if the specified I2C interrupt source is enabled or disabled.

**Parameters:** \_\_HANDLE\_\_: specifies the I2C Handle. This parameter can be I2Cx where x: 1 or 2 to select the I2C peripheral. \_\_INTERRUPT\_\_: 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:** Checks whether the specified I2C flag is set or not.

**Parameters:** \_\_HANDLE\_\_: specifies the I2C Handle. This parameter can be I2Cx where x: 1 or 2 to select the I2C peripheral. \_\_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:** Clears the I2C pending flags which are cleared by writing 0 in a specific bit.

**Parameters:** \_\_HANDLE\_\_: specifies the I2C Handle. This parameter can be I2Cx where x: 1 or 2 to select the I2C peripheral. \_\_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. This parameter can be I2Cx where x: 1 or 2 to select the I2C peripheral.

**Return value:**None:

- **\_\_HAL\_I2C\_CLEAR\_STOPFLAG**

**Description:** Clears the I2C STOPF pending flag.

**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\_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\_FLAG\_MASK

*I2C\_general\_call\_addressing\_mode*

- I2C\_GENERALCALL\_DISABLED
- I2C\_GENERALCALL\_ENABLED
- IS\_I2C\_GENERAL\_CALL

*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
- IS\_I2C\_MEMADD\_SIZE

*I2C\_nostretch\_mode*

- I2C\_NOSTRETCH\_DISABLED
- I2C\_NOSTRETCH\_ENABLED
- IS\_I2C\_NO\_STRETCH

*I2C\_Own\_Address1\_definition*

- IS\_I2C\_OWN\_ADDRESS1

*I2C\_Own\_Address2\_definition*

- IS\_I2C\_OWN\_ADDRESS2

*I2C Private Constants*

- I2C\_TIMEOUT\_FLAG
- I2C\_TIMEOUT\_ADDR\_SLAVE
- I2C\_MIN\_PCLK\_FREQ

*I2C Private Macros*

- I2C\_FREQRANGE
- I2C\_RISE\_TIME
- I2C\_SPEED\_STANDARD
- I2C\_SPEED\_FAST
- I2C\_SPEED
- I2C\_7BIT\_ADD\_WRITE
- I2C\_7BIT\_ADD\_READ
- I2C\_10BIT\_ADDRESS
- I2C\_10BIT\_HEADER\_WRITE
- I2C\_10BIT\_HEADER\_READ
- I2C\_MEM\_ADD\_MSB
- I2C\_MEM\_ADD\_LSB

## 22 HAL I2S Generic Driver

### 22.1 I2S Firmware driver registers structures

#### 22.1.1 I2S\_InitTypeDef

*I2S\_InitTypeDef* is defined in the `stm32l1xx_hal_i2s.h`

##### 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](#)

#### 22.1.2 I2S\_HandleTypeDef

*I2S\_HandleTypeDef* is defined in the `stm32l1xx_hal_i2s.h`

##### 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 HAL_I2S_ErrorTypeDef 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 HAL_I2S_ErrorTypeDef I2S_HandleTypeDef::ErrorCode`

## 22.2 I2S Firmware driver API description

The following section lists the various functions of the I2S library.

### 22.2.1 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

## 22.2.2 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.
- `HAL_I2S_Init()`
- `HAL_I2S_DelInit()`
- `HAL_I2S_MspInit()`
- `HAL_I2S_MspDelInit()`

## 22.2.3 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()
- ***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()***

## 22.2.4 Peripheral State and Errors functions

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

- ***HAL\_I2S\_GetState()***
- ***HAL\_I2S\_GetError()***

## 22.2.5 HAL\_I2S\_Init

|                      |                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_Init ( <i>I2S_HandleTypeDef</i> * <b>hi2s</b>)</b>                                                                                              |
| Function Description | Initializes the I2S according to the specified parameters in the <i>I2S_InitTypeDef</i> and create the associated handle.                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a <i>I2S_HandleTypeDef</i> structure that contains the configuration information for I2S module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                    |

## 22.2.6 HAL\_I2S\_DeInit

|               |                                                                                   |
|---------------|-----------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_I2S_DeInit ( <i>I2S_HandleTypeDef</i> * <b>hi2s</b>)</b> |
|---------------|-----------------------------------------------------------------------------------|

---

|                      |                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Deinitializes the I2S peripheral.                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                             |

## 22.2.7 HAL\_I2S\_MspInit

|                      |                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2S_MspInit ( <i>I2S_HandleTypeDef</i> * hi2s)</b>                                                                                                        |
| Function Description | I2S MSP Init.                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                             |

## 22.2.8 HAL\_I2S\_MspDelInit

|                      |                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2S_MspDelInit ( <i>I2S_HandleTypeDef</i> * hi2s)</b>                                                                                                     |
| Function Description | I2S MSP DelInit.                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                             |

## 22.2.9 HAL\_I2S\_Transmit

|               |                                                                      |
|---------------|----------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_I2S_Transmit ( <i>I2S_HandleTypeDef</i></b> |
|---------------|----------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <b>* hi2s, uint16_t * pData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Function Description | Transmit an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> <li>• <b>pData</b> : a 16-bit pointer to data buffer.</li> <li>• <b>Size</b> : number of data sample to be sent:</li> </ul>                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Timeout</b> : Timeout duration</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• When a 16-bit data frame or a 16-bit data frame extended is selected during the I2S configuration phase, the <b>Size</b> 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 <b>Size</b> parameter means the number of 16-bit data length.</li> <li>• The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).</li> </ul> |

## 22.2.10 HAL\_I2S\_Receive

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_Receive ( I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Receive an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> <li>• <b>pData</b> : a 16-bit pointer to data buffer.</li> <li>• <b>Size</b> : number of data sample to be sent:</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Timeout</b> : Timeout duration</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• When a 16-bit data frame or a 16-bit data frame extended is selected during the I2S configuration phase, the <b>Size</b> 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 <b>Size</b> parameter means the number of 16-bit data length.</li> <li>• The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).</li> <li>• In I2S Master Receiver mode, just after enabling the peripheral the clock will be generate in continouse way and as the I2S is not disabled at the end of the I2S transaction.</li> </ul> |

## 22.2.11 HAL\_I2S\_Transmit\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_Transmit_IT (</b><br><b>I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size)</b>                                                                                                                                                                                                                                                                                                                                                                                                         |
| Function Description | Transmit an amount of data in non-blocking mode with Interrupt.                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> <li>• <b>pData</b> : a 16-bit pointer to data buffer.</li> <li>• <b>Size</b> : number of data sample to be sent:</li> </ul>                                                                                                                                                                                                                                 |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> <li>• The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).</li> </ul> |

## 22.2.12 HAL\_I2S\_Receive\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_Receive_IT (</b><br><b>I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size)</b>                                                                                                                                                                                                                                                                                                                             |
| Function Description | Receive an amount of data in non-blocking mode with Interrupt.                                                                                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> <li>• <b>pData</b> : a 16-bit pointer to the Receive data buffer.</li> <li>• <b>Size</b> : number of data sample to be sent:</li> </ul>                                                                                                                                        |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> <li>• The I2S is kept enabled at the end of transaction to avoid the</li> </ul> |

- clock de-synchronization between Master and Slave(example: audio streaming).
- It is recommended to use DMA for the I2S receiver to avoid de-synchronisation between Master and Slave otherwise the I2S interrupt should be optimized.

### 22.2.13 HAL\_I2S\_Transmit\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_Transmit_DMA (</b><br><b>I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size)</b>                                                                                                                                                                                                                                                                                                                                                                                                        |
| Function Description | Transmit an amount of data in non-blocking mode with DMA.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> <li>• <b>pData</b> : a 16-bit pointer to the Transmit data buffer.</li> <li>• <b>Size</b> : number of data sample to be sent:</li> </ul>                                                                                                                                                                                                                    |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> <li>• The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).</li> </ul> |

### 22.2.14 HAL\_I2S\_Receive\_DMA

|                      |                                                                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_Receive_DMA (</b><br><b>I2S_HandleTypeDef * hi2s, uint16_t * pData, uint16_t Size)</b>                                                                                                                                                                                     |
| Function Description | Receive an amount of data in non-blocking mode with DMA.                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> <li>• <b>pData</b> : a 16-bit pointer to the Receive data buffer.</li> <li>• <b>Size</b> : number of data sample to be sent:</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                   |

---

|       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Notes | <ul style="list-style-type: none"> <li>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.</li> <li>The I2S is kept enabled at the end of transaction to avoid the clock de-synchronization between Master and Slave(example: audio streaming).</li> </ul> |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 22.2.15 HAL\_I2S\_DMAPause

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_DMAPause (</b><br><b>I2S_HandleTypeDef * hi2s)</b>                                                                                     |
| Function Description | Pauses the audio stream playing from the Media.                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li><b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                             |

## 22.2.16 HAL\_I2S\_DMAResume

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_DMAResume (</b><br><b>I2S_HandleTypeDef * hi2s)</b>                                                                                    |
| Function Description | Resumes the audio stream playing from the Media.                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li><b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                             |

## 22.2.17 HAL\_I2S\_DMAMainStop

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_I2S_DMAMainStop ( <i>I2S_HandleTypeDef</i> * <i>hi2s</i>)</b>                                                                              |
| Function Description | Resumes the audio stream playing from the Media.                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

## 22.2.18 HAL\_I2S\_IRQHandler

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2S_IRQHandler ( <i>I2S_HandleTypeDef</i> * <i>hi2s</i>)</b>                                                                                            |
| Function Description | This function handles I2S interrupt request.                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

## 22.2.19 HAL\_I2S\_TxHalfCpltCallback

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2S_TxHalfCpltCallback ( <i>I2S_HandleTypeDef</i> * <i>hi2s</i>)</b>                                                                                    |
| Function Description | Tx Transfer Half completed callbacks.                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 22.2.20 HAL\_I2S\_TxCpltCallback

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2S_TxCpltCallback ( <i>I2S_HandleTypeDef</i> * hi2s)</b>                                                                                               |
| Function Description | Tx Transfer completed callbacks.                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 22.2.21 HAL\_I2S\_RxHalfCpltCallback

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2S_RxHalfCpltCallback ( <i>I2S_HandleTypeDef</i> * hi2s)</b>                                                                                           |
| Function Description | Rx Transfer half completed callbacks.                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 22.2.22 HAL\_I2S\_RxCpltCallback

|                      |                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2S_RxCpltCallback ( <i>I2S_HandleTypeDef</i> * hi2s)</b>                                                                                               |
| Function Description | Rx Transfer completed callbacks.                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a I2S_HandleTypeDef structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                             |

### 22.2.23 HAL\_I2S\_ErrorCallback

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_I2S_ErrorCallback ( <i>I2S_HandleTypeDef</i> * hi2s)</b>                                                                                                       |
| Function Description | I2S error callbacks.                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a <i>I2S_HandleTypeDef</i> structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |

### 22.2.24 HAL\_I2S\_GetState

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_I2S_StateTypeDef HAL_I2S_GetState ( <i>I2S_HandleTypeDef</i> * hi2s)</b>                                                                                            |
| Function Description | Return the I2S state.                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a <i>I2S_HandleTypeDef</i> structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                                                                                         |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |

### 22.2.25 HAL\_I2S\_GetError

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_I2S_ErrorTypeDef HAL_I2S_GetError ( <i>I2S_HandleTypeDef</i> * hi2s)</b>                                                                                            |
| Function Description | Return the I2S error code.                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hi2s</b> : pointer to a <i>I2S_HandleTypeDef</i> structure that contains the configuration information for I2S module</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>I2S Error Code</b></li></ul>                                                                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                    |

## 22.3 I2S Firmware driver defines

### 22.3.1 I2S

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 Exported Macros*

- \_\_HAL\_I2S\_RESET\_HANDLE\_STATE

**Description:** Reset I2S handle state.

**Parameters:** \_\_HANDLE\_\_: specifies the I2S Handle.

**Return value:**None

- \_\_HAL\_I2S\_ENABLE

**Description:** 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_x` 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).
- **`_HAL_I2S_GET_FLAG`**  
**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 Legacy*

- `I2S_STANDARD_PHILLIPS`

#### *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**

## 23 HAL IRDA Generic Driver

### 23.1 IRDA Firmware driver registers structures

#### 23.1.1 IRDA\_InitTypeDef

*IRDA\_InitTypeDef* is defined in the `stm32l1xx_hal_irda.h`

##### 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:
  - IntegerDivider = ((PCLKx) / (8 \* (hirda->Init.BaudRate)))
  - FractionalDivider = ((IntegerDivider - ((uint32\_t) IntegerDivider)) \* 8) + 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
- ***uint32\_t IRDA\_InitTypeDef::IrDAMode*** Specifies the IrDA mode. This parameter can be a value of [\*IRDA\\_Low\\_Power\*](#)

#### 23.1.2 IRDA\_HandleTypeDef

*IRDA\_HandleTypeDef* is defined in the `stm32l1xx_hal_irda.h`

##### 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 HAL\_IRDA\_ErrorTypeDef 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 HAL\_IRDA\_ErrorTypeDef IRDA\_HandleTypeDef::ErrorCode** IRDA Error code

## 23.2 IRDA Firmware driver API description

The following section lists the various functions of the IRDA library.

### 23.2.1 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 these 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.
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



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

### 23.2.2 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: If the parity is enabled, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit. Depending on the frame length defined by the M bit (8-bits or 9-bits), the possible IRDA frame formats are as listed in the below table:
  - 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.

**Table 18: Frame formats**

| M bit | PCE bit | IRDA frame                 |
|-------|---------|----------------------------|
| 0     | 0       | SB   8 bit data   STB      |
| 0     | 1       | SB   7 bit data   PB   STB |
| 1     | 0       | SB   9 bit data   STB      |
| 1     | 1       | SB   8 bit data   PB   STB |

The HAL\_IRDA\_Init() function follows IRDA configuration procedures (details for the procedures are available in reference manual (RM0038)).

- [\*\*HAL\\_IRDA\\_Init\(\)\*\*](#)
- [\*\*HAL\\_IRDA\\_DeInit\(\)\*\*](#)
- [\*\*HAL\\_IRDA\\_MspInit\(\)\*\*](#)
- [\*\*HAL\\_IRDA\\_MspDeInit\(\)\*\*](#)

### 23.2.3 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()
  - ***HAL\_IRDA\_Transmit()***
  - ***HAL\_IRDA\_Receive()***
  - ***HAL\_IRDA\_Transmit\_IT()***
  - ***HAL\_IRDA\_Receive\_IT()***
  - ***HAL\_IRDA\_Transmit\_DMA()***
  - ***HAL\_IRDA\_Receive\_DMA()***
  - ***HAL\_IRDA\_DMAPause()***
  - ***HAL\_IRDA\_DMAResume()***
  - ***HAL\_IRDA\_DMAStop()***
  - ***HAL\_IRDA\_IRQHandler()***
  - ***HAL\_IRDA\_TxCpltCallback()***
  - ***HAL\_IRDA\_TxHalfCpltCallback()***
  - ***HAL\_IRDA\_RxCpltCallback()***
  - ***HAL\_IRDA\_RxHalfCpltCallback()***
  - ***HAL\_IRDA\_ErrorCallback()***

### 23.2.4 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.
- ***HAL\_IRDA\_GetState()***

- [\*HAL\\_IRDA\\_GetError\(\)\*](#)

### 23.2.5 HAL\_IRDA\_Init

|                      |                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_Init ( <i>IRDA_HandleTypeDef</i> * <i>hirda</i>)</b>                                                                                                                    |
| Function Description | Initializes the IRDA mode according to the specified parameters in the <i>IRDA_InitTypeDef</i> and create the associated handle.                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b><i>hirda</i></b> : Pointer to a <i>IRDA_HandleTypeDef</i> structure that contains the configuration information for the specified IRDA module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                             |

### 23.2.6 HAL\_IRDA\_DelInit

|                      |                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_DelInit ( <i>IRDA_HandleTypeDef</i> * <i>hirda</i>)</b>                                                                                                                 |
| Function Description | Deinitializes the IRDA peripheral.                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b><i>hirda</i></b> : Pointer to a <i>IRDA_HandleTypeDef</i> structure that contains the configuration information for the specified IRDA module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                             |

### 23.2.7 HAL\_IRDA\_MsplInit

|                      |                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_IRDA_MsplInit ( <i>IRDA_HandleTypeDef</i> * <i>hirda</i>)</b>                                                                                                                             |
| Function Description | IRDA MSP Init.                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b><i>hirda</i></b> : Pointer to a <i>IRDA_HandleTypeDef</i> structure that contains the configuration information for the specified IRDA module.</li> </ul> |

---

|               |                                                       |
|---------------|-------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>None.</li></ul> |
| Notes         | <ul style="list-style-type: none"><li>None.</li></ul> |

### 23.2.8 HAL\_IRDA\_MspDeInit

|                      |                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_IRDA_MspDeInit ( <i>IRDA_HandleTypeDef</i> * hirda)</b>                                                                                                                 |
| Function Description | IRDA MSP DeInit.                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li><b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                               |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                               |

### 23.2.9 HAL\_IRDA\_Transmit

|                      |                                                                                                                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_Transmit ( <i>IRDA_HandleTypeDef</i> * hirda, uint8_t * pData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                                |
| Function Description | Sends an amount of data in blocking mode.                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li><b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li><li><b>pData</b> : Pointer to data buffer</li><li><b>Size</b> : Amount of data to be sent</li><li><b>Timeout</b> : Specify timeout value</li></ul> |
| Return values        | <ul style="list-style-type: none"><li><b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                                                                            |

### 23.2.10 HAL\_IRDA\_Receive

|                      |                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_Receive (</b><br><b><i>IRDA_HandleTypeDef</i> * hirda, uint8_t * pData, uint16_t Size,</b><br><b>uint32_t Timeout)</b>                                                                                                                                                                                              |
| Function Description | Receive an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> <li>• <b>Timeout</b> : Specify timeout value</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                         |

### 23.2.11 HAL\_IRDA\_Transmit\_IT

|                      |                                                                                                                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_Transmit_IT (</b><br><b><i>IRDA_HandleTypeDef</i> * hirda, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                                |
| Function Description | Sends an amount of data in non-blocking mode.                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                   |

### 23.2.12 HAL\_IRDA\_Receive\_IT

|                      |                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_Receive_IT (</b><br><b><i>IRDA_HandleTypeDef</i> * hirda, uint8_t * pData, uint16_t Size)</b>                                                             |
| Function Description | Receives an amount of data in non-blocking mode.                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li> </ul> |

- |               |                                                                                                                                                  |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> <li>• None.</li> </ul>                                                           |

### 23.2.13 HAL\_IRDA\_Transmit\_DMA

|                      |                                                                                                                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_Transmit_DMA (</b><br><b><i>IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size</i></b> )                                                                                                                                                              |
| Function Description | Sends an amount of data in non-blocking mode.                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                   |

### 23.2.14 HAL\_IRDA\_Receive\_DMA

|                      |                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_Receive_DMA (</b><br><b><i>IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size</i></b> )                                                                                                                                                                   |
| Function Description | Receive an amount of data in non-blocking mode.                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• When the IRDA parity is enabled (PCE = 1) the data received contain the parity bit.</li> </ul>                                                                                                                                                         |

### 23.2.15 HAL\_IRDA\_DMAPause

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_DMAPause (</b><br><i>IRDA_HandleTypeDef * hirda)</i>                                                                                                    |
| Function Description | Pauses the DMA Transfer.                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                               |

### 23.2.16 HAL\_IRDA\_DMAResume

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_DMAResume (</b><br><i>IRDA_HandleTypeDef * hirda)</i>                                                                                                   |
| Function Description | Resumes the DMA Transfer.                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                               |

### 23.2.17 HAL\_IRDA\_DMAStop

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IRDA_DMAStop (</b><br><i>IRDA_HandleTypeDef * hirda)</i>                                                                                                     |
| Function Description | Stops the DMA Transfer.                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                   |

## Notes

- None.

### 23.2.18 HAL\_IRDA\_IRQHandler

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_IRDA_IRQHandler ( <i>IRDA_HandleTypeDef</i> * hirda)</b>                                                                                                                         |
| Function Description | This function handles IRDA interrupt request.                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hirda</b> : Pointer to a <i>IRDA_HandleTypeDef</i> structure that contains the configuration information for the specified IRDA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |

### 23.2.19 HAL\_IRDA\_TxCpltCallback

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_IRDA_TxCpltCallback ( <i>IRDA_HandleTypeDef</i> * hirda)</b>                                                                                                                     |
| Function Description | Tx Transfer completed callbacks.                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hirda</b> : Pointer to a <i>IRDA_HandleTypeDef</i> structure that contains the configuration information for the specified IRDA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |

### 23.2.20 HAL\_IRDA\_TxHalfCpltCallback

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_IRDA_TxHalfCpltCallback ( <i>IRDA_HandleTypeDef</i> * hirda)</b> |
| Function Description | Tx Half Transfer completed callbacks.                                        |

---

|               |                                                                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                |

### 23.2.21 HAL\_IRDA\_RxCpltCallback

|                      |                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_IRDA_RxCpltCallback ( <i>IRDA_HandleTypeDef</i> * hirda)</code>                                                                                                          |
| Function Description | Rx Transfer completed callbacks.                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                               |

### 23.2.22 HAL\_IRDA\_RxHalfCpltCallback

|                      |                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_IRDA_RxHalfCpltCallback ( <i>IRDA_HandleTypeDef</i> * hirda)</code>                                                                                                      |
| Function Description | Rx Half Transfer complete callbacks.                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                               |

### 23.2.23 HAL\_IRDA\_ErrorCallback

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_IRDA_ErrorCallback ( <i>IRDA_HandleTypeDef</i> * hirda)</b>                                                                                                               |
| Function Description | IRDA error callbacks.                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                               |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                               |

### 23.2.24 HAL\_IRDA\_GetState

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_IRDA_StateTypeDef HAL_IRDA_GetState ( <i>IRDA_HandleTypeDef</i> * hirda)</b>                                                                                                   |
| Function Description | Returns the IRDA state.                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                                                                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                               |

### 23.2.25 HAL\_IRDA\_GetError

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_IRDA_GetError ( <i>IRDA_HandleTypeDef</i> * hirda)</b>                                                                                                                |
| Function Description | Return the IRDA error code.                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hirda</b> : Pointer to a IRDA_HandleTypeDef structure that contains the configuration information for the specified IRDA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>IRDA Error Code</b></li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                               |

## 23.3 IRDA Firmware driver defines

### 23.3.1 IRDA

IRDA

#### *IRDA Exported Macros*

- **\_HAL\_IRDA\_RESET\_HANDLE\_STATE**

**Description:** Reset IRDA handle state.

**Parameters:** `_HANDLE_`: specifies the IRDA 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\_IRDA\_FLUSH\_DRREGISTER**

**Description:** Flushes the IRDA DR register.

**Parameters:** `_HANDLE_`: specifies the USART 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).

- **\_HAL\_IRDA\_GET\_FLAG**

**Description:** Checks whether the specified IRDA flag is set or not.

**Parameters:** `_HANDLE_`: specifies the IRDA 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). `_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:** Clears the specified IRDA pending flag.

**Parameters:** `_HANDLE_`: specifies the IRDA 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). `_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:

- **\_HAL\_IRDA\_CLEAR\_PFLAG**

**Description:** Clear the IRDA PE pending flag.

**Parameters:** `_HANDLE_`: specifies the IRDA 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\_IRDA\_CLEAR\_FEFLAG**

**Description:** Clear the IRDA FE pending flag.

**Parameters:** `_HANDLE_`: specifies the IRDA 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\_IRDA\_CLEAR\_NEFLAG**

**Description:** Clear the IRDA NE pending flag.

**Parameters:** `_HANDLE_`: specifies the IRDA 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\_IRDA\_CLEAR\_OREFLAG**

**Description:** Clear the IRDA ORE pending flag.

**Parameters:** `_HANDLE_`: specifies the IRDA 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\_IRDA\_CLEAR\_IDLEFLAG**

**Description:** Clear the IRDA IDLE pending flag.

**Parameters:** `_HANDLE_`: specifies the IRDA 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\_IRDA\_ENABLE\_IT**

**Description:** Enables the specified IRDA interrupt.

**Parameters:** `_HANDLE_`: specifies the IRDA 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). `_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**

**Description:** Disables the specified IRDA interrupt.

**Parameters:** `_HANDLE_`: specifies the IRDA 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). `_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:** Checks whether the specified IRDA interrupt has occurred or not.

**Parameters:** `_HANDLE_`: specifies the IRDA 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). `_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\_ENABLE**

**Description:** Enable USART/USART associated to IRDA Handle.

**Parameters:** `_HANDLE_`: specifies the IRDA 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\_IRDA\_DISABLE**

**Description:** Disable USART/USART associated to IRDA Handle.

**Parameters:** `__HANDLE__`: specifies the IRDA 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:

*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 interruptions flag mask*

- `IRDA_IT_MASK`

*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`
- `IS_IRDA_POWERMODE`

*IRDA Parity*

- `IRDA_PARITY_NONE`
- `IRDA_PARITY_EVEN`
- `IRDA_PARITY_ODD`
- `IS_IRDA_PARITY`

*IRDA Private Constants*

- `IRDA_TIMEOUT_VALUE`
- `IRDA_DR_MASK_U16_7DATABITS`
- `IRDA_DR_MASK_U16_8DATABITS`
- `IRDA_DR_MASK_U16_9DATABITS`
- `IRDA_DR_MASK_U8_7DATABITS`
- `IRDA_DR_MASK_U8_8DATABITS`

*IRDA Private Macros*

- `IRDA_DIV`
- `IRDA_DIVMANT`
- `IRDA_DIVFRAQ`

- **IRDA\_BRR**

- **IS\_IRDA\_BAUDRATE**

**Description:** Ensure that IRDA Baud rate is less or equal to maximum value.

**Parameters:** BAUDRATE: specifies the IRDA Baudrate set by the user.

**Return value:**True: or False

#### *IRDA Transfer Mode*

- **IRDA\_MODE\_RX**

- **IRDA\_MODE\_TX**

- **IRDA\_MODE\_TX\_RX**

- **IS\_IRDA\_MODE**

#### *IRDA Word Length*

- **IRDA\_WORDLENGTH\_8B**

- **IRDA\_WORDLENGTH\_9B**

- **IS\_IRDA\_WORD\_LENGTH**

## 24 HAL IWDG Generic Driver

### 24.1 IWDG Firmware driver registers structures

#### 24.1.1 IWDG\_InitTypeDef

*IWDG\_InitTypeDef* is defined in the `stm32l1xx_hal_iwdg.h`

##### 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 = 0xFFFF

#### 24.1.2 IWDG\_HandleTypeDefDef

*IWDG\_HandleTypeDefDef* is defined in the `stm32l1xx_hal_iwdg.h`

##### Data Fields

- *IWDG\_TypeDef \* Instance*
- *IWDG\_InitTypeDef Init*
- *HAL\_LockTypeDef Lock*
- *\_\_IO HAL\_IWDG\_StateTypeDef State*

##### Field Documentation

- *IWDG\_TypeDef\* IWDG\_HandleTypeDefDef::Instance* Register base address
- *IWDG\_InitTypeDef IWDG\_HandleTypeDefDef::Init* IWDG required parameters
- *HAL\_LockTypeDef IWDG\_HandleTypeDefDef::Lock* IWDG Locking object
- *\_\_IO HAL\_IWDG\_StateTypeDef IWDG\_HandleTypeDefDef::State* IWDG communication state

### 24.2 IWDG Firmware driver API description

The following section lists the various functions of the IWDG library.

#### 24.2.1 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize the IWDG according to the specified parameters in the *IWDG\_InitTypeDef* and create the associated handle
- Initialize the IWDG MSP

- DeInitialize IWDG MSP
- [\*HAL\\_IWDG\\_Init\(\)\*](#)
- [\*HAL\\_IWDG\\_MspInit\(\)\*](#)

## 24.2.2 IO operation functions

This section provides functions allowing to:

- Start the IWDG.
- Refresh the IWDG.
- [\*HAL\\_IWDG\\_Start\(\)\*](#)
- [\*HAL\\_IWDG\\_Refresh\(\)\*](#)

## 24.2.3 Peripheral State functions

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

- [\*HAL\\_IWDG\\_GetState\(\)\*](#)

## 24.2.4 HAL\_IWDG\_Init

|                      |                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IWDG_Init ( <i>IWDG_HandleTypeDef</i> * <i>hiwdg</i>)</b>                                                                                                             |
| Function Description | Initializes the IWDG according to the specified parameters in the <i>IWDG_InitTypeDef</i> and creates the associated handle.                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hiwdg</b> : pointer to a <i>IWDG_HandleTypeDef</i> structure that contains the configuration information for the specified IWDG module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                      |

## 24.2.5 HAL\_IWDG\_MspInit

|                      |                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_IWDG_MspInit ( <i>IWDG_HandleTypeDef</i> * <i>hiwdg</i>)</b>                                                                                                                       |
| Function Description | Initializes the IWDG MSP.                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hiwdg</b> : pointer to a <i>IWDG_HandleTypeDef</i> structure that contains the configuration information for the specified IWDG module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                      |

## Notes

- None.

### 24.2.6 HAL\_IWDG\_Start

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IWDG_Start ( <i>IWDG_HandleTypeDef</i> * <i>hiwdg</i>)</b>                                                                                                   |
| Function Description | Starts the IWDG.                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hiwdg</b> : pointer to a IWDG_HandleTypeDef structure that contains the configuration information for the specified IWDG module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                               |

### 24.2.7 HAL\_IWDG\_Refresh

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_IWDG_Refresh ( <i>IWDG_HandleTypeDef</i> * <i>hiwdg</i>)</b>                                                                                                 |
| Function Description | Refreshes the IWDG.                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hiwdg</b> : pointer to a IWDG_HandleTypeDef structure that contains the configuration information for the specified IWDG module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                               |

### 24.2.8 HAL\_IWDG\_GetState

|               |                                                                                            |
|---------------|--------------------------------------------------------------------------------------------|
| Function Name | <b>HAL_IWDG_StateTypeDef HAL_IWDG_GetState ( <i>IWDG_HandleTypeDef</i> * <i>hiwdg</i>)</b> |
|---------------|--------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Returns the IWDG state.                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hiwdg</b> : pointer to a IWDG_HandleTypeDef structure that contains the configuration information for the specified IWDG module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                               |

## 24.3 IWDG Firmware driver defines

### 24.3.1 IWDG

IWDG

#### *IWDG Exported Macros*

- **\_HAL\_IWDG\_RESET\_HANDLE\_STATE**  
**Description:** Reset IWDG handle state.  
**Parameters:** \_\_HANDLE\_\_: IWDG handle.  
**Return value:**None;
- **\_HAL\_IWDG\_START**  
**Description:** Enables the IWDG peripheral.  
**Parameters:** \_\_HANDLE\_\_: IWDG handle  
**Return value:**None;
- **\_HAL\_IWDG\_RELOAD\_COUNTER**  
**Description:** Reloads IWDG counter with value defined in the reload register (write access to IWDG\_PR and IWDG\_RLR registers disabled).  
**Parameters:** \_\_HANDLE\_\_: IWDG handle  
**Return value:**None;
- **\_HAL\_IWDG\_ENABLE\_WRITE\_ACCESS**  
**Description:** Enables write access to IWDG\_PR and IWDG\_RLR registers.  
**Parameters:** \_\_HANDLE\_\_: IWDG handle  
**Return value:**None;
- **\_HAL\_IWDG\_DISABLE\_WRITE\_ACCESS**  
**Description:** Disables write access to IWDG\_PR and IWDG\_RLR registers.  
**Parameters:** \_\_HANDLE\_\_: IWDG handle  
**Return value:**None;
- **\_HAL\_IWDG\_GET\_FLAG**  
**Description:** Gets the selected IWDG's flag status.  
**Parameters:** \_\_HANDLE\_\_: IWDG handle \_\_FLAG\_\_: specifies the flag to check. This parameter can be one of the following values: IWDG\_FLAG\_PVU: Watchdog counter reload value update flag IWDG\_FLAG\_RVU: Watchdog counter prescaler value flag  
**Return value:**The new state of \_\_FLAG\_\_ (TRUE or FALSE).

#### *IWDG\_Flag\_definition*

- **IWDG\_FLAG\_PVU**  
Watchdog counter prescaler value update Flag
- **IWDG\_FLAG\_RVU**  
Watchdog counter reload value update Flag

#### *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
- **IS\_IWDG\_PRESCALER**

*IWDG Private Defines*

- **HAL\_IWDG\_DEFAULT\_TIMEOUT**

*IWDG\_Registers\_BitMask*

- **KR\_KEY\_RELOAD**  
IWDG Reload Counter Enable
- **KR\_KEY\_ENABLE**  
IWDG Peripheral Enable
- **KR\_KEY\_EWA**  
IWDG KR Write Access Enable
- **KR\_KEY\_DWA**  
IWDG KR Write Access Disable
- **IS\_IWDG\_KR**

*IWDG\_Reload\_Value*

- **IS\_IWDG\_RELOAD**

## 25 HAL LCD Generic Driver

### 25.1 LCD Firmware driver registers structures

#### 25.1.1 LCD\_InitTypeDef

*LCD\_InitTypeDef* is defined in the `stm32l1xx_hal_lcd.h`

##### 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 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::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.

#### 25.1.2 LCD\_HandleTypeDef

*LCD\_HandleTypeDef* is defined in the `stm32l1xx_hal_lcd.h`

##### Data Fields

- *LCD\_TypeDef \* Instance*
- *LCD\_InitTypeDef Init*
- *HAL\_LockTypeDef Lock*
- *\_IO HAL\_LCD\_StateTypeDef State*
- *\_IO HAL\_LCD\_ErrorTypeDef 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 HAL\_LCD\_ErrorTypeDef LCD\_HandleTypeDef::ErrorCode*

## 25.2 LCD Firmware driver API description

The following section lists the various functions of the LCD library.

### 25.2.1 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.

The LCD HAL driver can be used as follows: (#) Declare a LCD\_HandleTypeDef handle structure. (#) Initialize the LCD low level resources by implement the HAL\_LCD\_MsplInit() API: (##) 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.
  - a. LCD pins configuration:
- Enable the clock for the LCD GPIOs.
- Configure these LCD pins as alternate function no-pull.
  - a. Enable the LCD interface clock. (#) Program the Prescaler, Divider, Blink mode, Blink Frequency Duty, Bias, Voltage Source, Dead Time, Pulse On Duration and Contrast in the hlcd Init structure. (#) 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\_MsplInit() API. -@- After calling the HAL\_LCD\_Init() the LCD RAM memory is cleared (#) 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 (#) 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. (#) The HAL\_LCD\_Clear() API can be used to clear the LCD RAM memory. (#) 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.

## 25.2.2 HAL\_LCD\_DeInit

|                      |                                                                                |
|----------------------|--------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_LCD_DeInit ( <i>LCD_HandleTypeDef</i> *<br/>hlcd)</b> |
| Function Description |                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                      |

## 25.2.3 HAL\_LCD\_Init

|                      |                                                                          |
|----------------------|--------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_LCD_Init ( <i>LCD_HandleTypeDef</i> * hlcd)</b> |
| Function Description |                                                                          |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                  |

#### 25.2.4 HAL\_LCD\_MspInit

|                      |                                                                |
|----------------------|----------------------------------------------------------------|
| Function Name        | <b>void HAL_LCD_MspInit ( <i>LCD_HandleTypeDef</i> * hlcd)</b> |
| Function Description |                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>        |

#### 25.2.5 HAL\_LCD\_MspDeInit

|                      |                                                                  |
|----------------------|------------------------------------------------------------------|
| Function Name        | <b>void HAL_LCD_MspDeInit ( <i>LCD_HandleTypeDef</i> * hlcd)</b> |
| Function Description |                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>          |

#### 25.2.6 HAL\_LCD\_Write

|                      |                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_LCD_Write ( <i>LCD_HandleTypeDef</i> * hlcd, uint32_t RAMRegisterIndex, uint32_t RAMRegisterMask, uint32_t Data)</b> |
| Function Description |                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                       |

### 25.2.7 HAL\_LCD\_Clear

Function Name      **HAL\_StatusTypeDef HAL\_LCD\_Clear ( *LCD\_HandleTypeDef* \* hlcd)**

Function Description

Notes                •     None.

### 25.2.8 HAL\_LCD\_UpdateDisplayRequest

Function Name      **HAL\_StatusTypeDef HAL\_LCD\_UpdateDisplayRequest ( *LCD\_HandleTypeDef* \* hlcd)**

Function Description

Notes                •     None.

### 25.2.9 HAL\_LCD\_GetState

Function Name      **HAL\_LCD\_StateTypeDef HAL\_LCD\_GetState ( *LCD\_HandleTypeDef* \* hlcd)**

Function Description

Notes                •     None.

### 25.2.10 HAL\_LCD\_GetError

Function Name      **uint32\_t HAL\_LCD\_GetError ( *LCD\_HandleTypeDef* \* hlcd)**

Function Description

Notes                •     None.

## 25.3 LCD Firmware driver defines

### 25.3.1 LCD

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 Exported Macros**

- **\_\_HAL\_LCD\_RESET\_HANDLE\_STATE**  
**Description:** Reset LCD handle state.  
**Parameters:** \_\_HANDLE\_\_: specifies the LCD Handle.  
**Return value:**None
- **\_\_HAL\_LCD\_ENABLE**  
**Description:** macros to enables or disables the LCD  
**Parameters:** \_\_HANDLE\_\_: specifies the LCD Handle.  
**Return value:**None
- **\_\_HAL\_LCD\_DISABLE**
- **\_\_HAL\_LCD\_HIGHDRIVER\_ENABLE**  
**Description:** Macros to enable or disable the low resistance divider.  
**Parameters:** \_\_HANDLE\_\_: specifies the LCD Handle.  
**Return value:**None
- **\_\_HAL\_LCD\_HIGHDRIVER\_DISABLE**
- **\_\_HAL\_LCD\_PULSEONDURATION\_CONFIG**  
**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 = 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

**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).

- **\_\_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\_EWS**: LCD Enabled flag. It indicates the LCD controller status.

**Return value**: The new state of **\_\_FLAG\_\_** (TRUE or FALSE).

- **\_\_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\_EWS**
- **LCD\_FLAG\_SOF**
- **LCD\_FLAG\_UDR**
- **LCD\_FLAG\_UDD**
- **LCD\_FLAG\_RDY**
- **LCD\_FLAG\_FCRSF**

#### **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**

## 26 HAL NOR Generic Driver

### 26.1 NOR Firmware driver registers structures

#### 26.1.1 NOR\_IDTypeDef

*NOR\_IDTypeDef* is defined in the `stm32l1xx_hal_nor.h`

##### Data Fields

- `uint16_t ManufacturerCode`
- `uint16_t DeviceCode1`
- `uint16_t DeviceCode2`
- `uint16_t DeviceCode3`

##### Field Documentation

- `uint16_t NOR_IDTypeDef::ManufacturerCode` Defines the device's manufacturer code used to identify the memory
- `uint16_t NOR_IDTypeDef::DeviceCode1`
- `uint16_t NOR_IDTypeDef::DeviceCode2`
- `uint16_t NOR_IDTypeDef::DeviceCode3` Defines the devices' 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

#### 26.1.2 NOR\_CFITypeDef

*NOR\_CFITypeDef* is defined in the `stm32l1xx_hal_nor.h`

##### Data Fields

- `uint16_t CFI1`
- `uint16_t CFI2`
- `uint16_t CFI3`
- `uint16_t CFI4`

##### Field Documentation

- `uint16_t NOR_CFITypeDef::CFI1` < 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::CFI2`
- `uint16_t NOR_CFITypeDef::CFI3`
- `uint16_t NOR_CFITypeDef::CFI4`

#### 26.1.3 NOR\_HandleTypeDef

*NOR\_HandleTypeDef* is defined in the `stm32l1xx_hal_nor.h`

##### 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

## 26.2 NOR Firmware driver API description

The following section lists the various functions of the NOR library.

### 26.2.1 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

## 26.2.2 NOR Initialization and de\_initialization functions

This section provides functions allowing to initialize/de-initialize the NOR memory

- `HAL_NOR_Init()`
- `HAL_NOR_DelInit()`
- `HAL_NOR_MspInit()`
- `HAL_NOR_MspDelInit()`
- `HAL_NOR_MspWait()`

## 26.2.3 NOR Input and Output functions

This section provides functions allowing to use and control the NOR memory

- `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()`

## 26.2.4 NOR Control functions

This subsection provides a set of functions allowing to control dynamically the NOR interface.

- `HAL_NOR_WriteOperation_Enable()`
- `HAL_NOR_WriteOperation_Disable()`

## 26.2.5 NOR State functions

This subsection permits to get in run-time the status of the NOR controller and the data flow.

- `HAL_NOR_GetState()`
- `HAL_NOR_GetStatus()`

## 26.2.6 HAL\_NOR\_Init

Function Name      `HAL_StatusTypeDef HAL_NOR_Init ( NOR_HandleTypeDef * hnор, FSMC_NORSRAM_TimingTypeDef * Timing,`

**FSMC\_NORSRAM\_TimingTypeDef \* ExtTiming)**

|                      |                                                                                                                                                                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Perform the NOR memory Initialization sequence.                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>Timing</b> : pointer to NOR control timing structure</li> <li>• <b>ExtTiming</b> : pointer to NOR extended mode timing structure</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                   |
| Notes                | • None.                                                                                                                                                                                                                                                                                                               |

**26.2.7 HAL\_NOR\_DeInit**

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_DeInit ( NOR_HandleTypeDef * hnор)</b>                                                                                                    |
| Function Description | Perform NOR memory De-Initialization sequence.                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                    |

Notes • None.

**26.2.8 HAL\_NOR\_MspInit**

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NOR_MspInit ( NOR_HandleTypeDef * hnор)</b>                                                                                                                |
| Function Description | NOR MSP Init.                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> </ul> |
| Return values        | • None.                                                                                                                                                                |

Notes • None.

## 26.2.9 HAL\_NOR\_MspDeInit

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NOR_MspDeInit ( <i>NOR_HandleTypeDef</i> * <i>hnor</i>)</b>                                                                                                |
| Function Description | NOR MSP DeInit.                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                              |

## 26.2.10 HAL\_NOR\_MspWait

|                      |                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_NOR_MspWait ( <i>NOR_HandleTypeDef</i> * <i>hnor</i>,<br/>uint32_t <i>Timeout</i>)</b>                                                                                                                       |
| Function Description | NOR BSP Wait fro Ready/Busy signal.                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>Timeout</b> : Maximum timeout value</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                |

## 26.2.11 HAL\_NOR\_Read\_ID

|                      |                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_Read_ID ( <i>NOR_HandleTypeDef</i> * <i>hnor</i>, <i>NOR_IDTypeDef</i> * <i>pNOR_ID</i>)</b>                                                                                                      |
| Function Description | Read NOR flash IDs.                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>pNOR_ID</b> : pointer to NOR ID structure</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                      |

### 26.2.12 HAL\_NOR\_ReturnToReadMode

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_ReturnToReadMode ( <i>NOR_HandleTypeDef</i> * <i>hnor</i>)</b>                                                                            |
| Function Description | Returns the NOR memory to Read mode.                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                              |

### 26.2.13 HAL\_NOR\_Read

|                      |                                                                                                                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_Read ( <i>NOR_HandleTypeDef</i> * <i>hnor</i>, uint32_t * <i>pAddress</i>, uint16_t * <i>pData</i>)</b>                                                                                                                                         |
| Function Description | Read data from NOR memory.                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>pAddress</b> : pointer to Device address</li> <li>• <b>pData</b> : pointer to read data</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                    |

### 26.2.14 HAL\_NOR\_Program

|                      |                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_Program ( <i>NOR_HandleTypeDef</i> * <i>hnor</i>, uint32_t * <i>pAddress</i>, uint16_t * <i>pData</i>)</b> |
| Function Description | Program data to NOR memory.                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that</li> </ul>                         |

|                                               |                                                        |
|-----------------------------------------------|--------------------------------------------------------|
|                                               | contains the configuration information for NOR module. |
| • <b>pAddress</b> : Device address            |                                                        |
| • <b>pData</b> : pointer to the data to write |                                                        |
| Return values                                 | • <b>HAL status</b>                                    |

### 26.2.15 HAL\_NOR\_ReadBuffer

|                      |                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_ReadBuffer (</b><br><b>NOR_HandleTypeDef * hnор, uint32_t uwAddress, uint16_t * pData, uint32_t uwBufferSize)</b>                                                                                                                                                                                                                                                                |
| Function Description | Reads a block of data from the FSMC NOR memory.                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>uwAddress</b> : NOR memory internal address to read from.</li> <li>• <b>pData</b> : pointer to the buffer that receives the data read from the NOR memory.</li> <li>• <b>uwBufferSize</b> : number of Half word to read.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                           |

### 26.2.16 HAL\_NOR\_ProgramBuffer

|                      |                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_ProgramBuffer (</b><br><b>NOR_HandleTypeDef * hnор, uint32_t uwAddress, uint16_t * pData, uint32_t uwBufferSize)</b>                                                                                                                                                                                                                                                 |
| Function Description | Writes a half-word buffer to the FSMC NOR memory.                                                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>uwAddress</b> : NOR memory internal address from which the data</li> <li>• <b>pData</b> : pointer to source data buffer.</li> <li>• <b>uwBufferSize</b> : number of Half words to write. The maximum allowed</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                               |

## Notes

- None.

### 26.2.17 HAL\_NOR\_Erase\_Block

|                      |                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_Erase_Block (</b><br><b>NOR_HandleTypeDef * hnor, uint32_t BlockAddress, uint32_t Address)</b>                                                                                                                                                 |
| Function Description | Erase the specified block of the NOR memory.                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>BlockAddress</b> : : Block to erase address</li> <li>• <b>Address</b> : Device address</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                   |

### 26.2.18 HAL\_NOR\_Erase\_Chip

|                      |                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_Erase_Chip (</b><br><b>NOR_HandleTypeDef * hnor, uint32_t Address)</b>                                                                                                                 |
| Function Description | Erase the entire NOR chip.                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnor</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>Address</b> : : Device address</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                           |

### 26.2.19 HAL\_NOR\_Read\_CFI

|               |                                             |
|---------------|---------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_NOR_Read_CFI (</b> |
|---------------|---------------------------------------------|

**NOR\_HandleTypeDef \* hnор, NOR\_CFITypeDef \* pNOR\_CFI)**

|                      |                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Read NOR flash CFI IDs.                                                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li><li>• <b>pNOR_CFI</b> : pointer to NOR CFI IDs structure</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                           |

### 26.2.20 HAL\_NOR\_WriteOperation\_Enable

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_WriteOperation_Enable (</b><br><b>NOR_HandleTypeDef * hnор)</b>                                                                         |
| Function Description | Enables dynamically NOR write operation.                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 26.2.21 HAL\_NOR\_WriteOperation\_Disable

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_NOR_WriteOperation_Disable (</b><br><b>NOR_HandleTypeDef * hnор)</b>                                                                        |
| Function Description | Disables dynamically NOR write operation.                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

## 26.2.22 HAL\_NOR\_GetState

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_NOR_StateTypeDef HAL_NOR_GetState (</b><br><b>NOR_HandleTypeDef * hnор)</b>                                                                                     |
| Function Description | return the NOR controller state                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>NOR controller state</b></li> </ul>                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                              |

## 26.2.23 HAL\_NOR\_GetStatus

|                      |                                                                                                                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>NOR_StatusTypeDef HAL_NOR_GetStatus (</b><br><b>NOR_HandleTypeDef * hnор, uint32_t Address, uint32_t Timeout)</b>                                                                                                                                                 |
| Function Description | Returns the NOR operation status.                                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hnор</b> : pointer to a NOR_HandleTypeDef structure that contains the configuration information for NOR module.</li> <li>• <b>Address</b> : Device address</li> <li>• <b>Timeout</b> : NOR progamming Timeout</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>NOR_Status</b> : The returned value can be:<br/><b>NOR_SUCCESS, NOR_ERROR or NOR_TIMEOUT</b></li> </ul>                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                            |

## 26.3 NOR Firmware driver defines

### 26.3.1 NOR

NOR

***NOR Exported Constants***

- **MC\_ADDRESS**
- **DEVICE\_CODE1\_ADDR**
- **DEVICE\_CODE2\_ADDR**
- **DEVICE\_CODE3\_ADDR**
- **CFI1\_ADDRESS**

- **CFI2\_ADDRESS**
- **CFI3\_ADDRESS**
- **CFI4\_ADDRESS**
- **NOR\_TMEOUT**
- **NOR\_MEMORY\_8B**
- **NOR\_MEMORY\_16B**
- **NOR\_MEMORY\_ADDRESS1**
- **NOR\_MEMORY\_ADDRESS2**
- **NOR\_MEMORY\_ADDRESS3**
- **NOR\_MEMORY\_ADDRESS4**

*NOR Exported macro*

- **\_\_HAL\_NOR\_RESET\_HANDLE\_STATE**  
**Description:** Reset NOR handle state.  
**Parameters:** \_\_HANDLE\_\_: NOR handle  
**Return value:**None
- **\_\_NOR\_ADDR\_SHIFT**  
**Description:** NOR memory address shifting.  
**Parameters:** \_\_NOR\_ADDRESS\_\_: NOR base address \_\_NOR\_MEMORY\_WIDTH\_\_: NOR memory width \_\_ADDRESS\_\_: NOR memory address  
**Return value:**NOR: shifted address value
- **\_\_NOR\_WRITE**  
**Description:** NOR memory write data to specified address.  
**Parameters:** \_\_ADDRESS\_\_: NOR memory address \_\_DATA\_\_: Data to write  
**Return value:**None

## 27 HAL OPAMP Generic Driver

### 27.1 OPAMP Firmware driver registers structures

#### 27.1.1 OPAMP\_InitTypeDef

*OPAMP\_InitTypeDef* is defined in the `stm32l1xx_hal_opamp.h`

##### Data Fields

- *uint32\_t PowerSupplyRange*
- *uint32\_t UserTrimming*
- *uint32\_t Mode*
- *uint32\_t InvertingInput*
- *uint32\_t NonInvertingInput*
- *uint32\_t PowerMode*
- *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::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::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 mode
  - In Standalone mode: i.e when mode is OPAMP\_STANDALONE\_MODE This parameter must be a value of *OPAMP\_InvertingInput* InvertingInput is either VM0 or VM1
  - In 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* NonInvertingInput is either VP0, VP1 or VP2
- *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::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)
- *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)
- ***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)
- ***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)

### 27.1.2 OPAMP\_HandleTypeDef

***OPAMP\_HandleTypeDef*** is defined in the `stm32l1xx_hal_opamp.h`

#### 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

## 27.2 OPAMP Firmware driver API description

The following section lists the various functions of the OPAMP library.

### 27.2.1 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. 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 trimming search wait time.
3. 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.
  4. Running mode: Follower mode
    - No Inverting Input is connected.
    - The OPAMP(s) output(s) are internally connected to inverting input
  5. The OPAMPs inverting input can be selected among the list shown in table below.
  6. The OPAMPs non inverting input can be selected among the list shown in table below.

**Table 19: OPAMPs inverting/non-inverting inputs for STM32L1 devices**

|                                 | HAL parameter name                                         | OPAMP1                            | OPAMP2                            | OPAMP3 <sup>(1)</sup>             |
|---------------------------------|------------------------------------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|
| Inverting inputs <sup>(2)</sup> | VM0<br>VM1                                                 | PA2<br>VINM pin<br><sup>(3)</sup> | PA7<br>VINM pin<br><sup>(3)</sup> | PC2<br>VINM pin<br><sup>(3)</sup> |
| non-inverting inputs            | VP0<br>DAC_CH1 <sup>(4)</sup><br>DAC_CH2<br><sup>(4)</sup> | PA1<br>DAC_CH1<br>-               | PA6<br>DAC_CH1<br>DAC_CH2         | PC1<br>-<br>DAC_CH2               |

**Notes:**

<sup>(1)</sup> OPAMP3 availability depends on device category.

<sup>(2)</sup>NA in follower mode.

<sup>(3)</sup>OPAMP input OPAMPx\_VINM are dedicated OPAMP pins, their availability depends on device package.

<sup>(4)</sup> DAC channels 1 and 2 are connected internally to OPAMP. Nevertheless, I/O pins connected to DAC can still be used as DAC output (pins PA4 and PA5).

**Table 20: OPAMP outputs for STM32L1 devices**

|        | OPAMP1 | OPAMP2 | OPAMP3 <sup>(1)</sup> |
|--------|--------|--------|-----------------------|
| Output | PA3    | PB0    | PC3                   |

**Notes:**

<sup>(1)</sup>OPAMP3 availability depends on device category.

**27.2.2 How to use this driver**

1. Calibration: To run the opamp calibration self calibration
  - Start calibration using HAL\_OPAMP\_SelfCalibrate.

- Store the calibration results.
- 2. Running mode: To use the opamp, perform the following steps
  - Fill in the HAL\_OPAMP\_MspInit() to
    - Enable the OPAMP Peripheral clock using macro "`__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.
  - 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).
  - Enable the opamp using HAL\_OPAMP\_Start() function.
  - Disable the opamp using HAL\_OPAMP\_Stop() function.
  - 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)
  - If needed, unlock the opamp using HAL\_OPAMPEx\_Unlock() function.
- 3. Running mode: change of configuration while OPAMP ON (change on the fly)
  - If needed, Fill in the HAL\_OPAMP\_MspInit()
    - This is the case for instance if you wish to use new OPAMP I/O
  - Configure the opamp using HAL\_OPAMP\_Init() function:
    - As in configure case, selects first the parameters you wish to modify.

### 27.2.3 Initialization and de-initialization functions

This section provides functions allowing to:

- `HAL_OPAMP_Init()`
- `HAL_OPAMP_DelInit()`
- `HAL_OPAMP_MspInit()`
- `HAL_OPAMP_MspDelInit()`

### 27.2.4 IO operation functions

This subsection provides a set of functions allowing to manage the OPAMP start, stop and calibration actions.

- `HAL_OPAMP_GetTrimOffset()`

### 27.2.5 Peripheral Control functions

- `HAL_OPAMP_Lock()`

### 27.2.6 Peripheral State functions

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

- ***HAL\_OPAMP\_GetState()***

### 27.2.7 HAL\_OPAMP\_Init

|                      |                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_OPAMP_Init (</b><br><b><i>OPAMP_HandleTypeDef * hopamp</i></b> )                                                                                    |
| 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"> <li>• <b>hopamp</b> : OPAMP handle</li> </ul>                                                                                             |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• If the selected opamp is locked, initialization can't be performed. To unlock the configuration, perform a system reset.</li> </ul> |

### 27.2.8 HAL\_OPAMP\_DelInit

|                      |                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_OPAMP_DelInit (</b><br><b><i>OPAMP_HandleTypeDef * hopamp</i></b> )                                                                                       |
| Function Description | Deinitializes the OPAMP peripheral.                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hopamp</b> : OPAMP handle</li> </ul>                                                                                                   |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• Deinitialization can't be performed if the OPAMP configuration is locked. To unlock the configuration, perform a system reset.</li> </ul> |

### 27.2.9 HAL\_OPAMP\_MspInit

|                      |                                                                                  |
|----------------------|----------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_OPAMP_MspInit (</b><br><b><i>OPAMP_HandleTypeDef * hopamp</i></b> )  |
| Function Description | Initializes the OPAMP MSP.                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hopamp</b> : OPAMP handle</li> </ul> |

---

|               |                                                       |
|---------------|-------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>None.</li></ul> |
| Notes         | <ul style="list-style-type: none"><li>None.</li></ul> |

### 27.2.10 HAL\_OPAMP\_MspDeInit

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_OPAMP_MspDeInit ( <i>OPAMP_HandleTypeDef</i> *<br/>hopamp)</b>   |
| Function Description | Deinitializes OPAMP MSP.                                                     |
| Parameters           | <ul style="list-style-type: none"><li><b>hopamp</b> : OPAMP handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                        |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                        |

### 27.2.11 HAL\_OPAMP\_GetTrimOffset

|                      |                                                                                                                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>OPAMP_TrimmingValueTypeDef HAL_OPAMP_GetTrimOffset<br/>( <i>OPAMP_HandleTypeDef</i> * hopamp, uint32_t trimmingoffset)</b>                                                                                                                                                                      |
| 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"><li><b>hopamp</b> : OPAMP handle</li><li><b>trimmingoffset</b> : Trimming offset (P or N) This parameter must be a value of OPAMP FactoryTrimming</li></ul>                                                                                                      |
| Return values        | <ul style="list-style-type: none"><li><b>Trimming value (P or N): range: 0-&gt;31 or<br/>OPAMP_FACTORYTRIMMING_DUMMY if trimming value is not available</b></li></ul>                                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>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.</li></ul> |

### 27.2.12 HAL\_OPAMP\_Lock

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_OPAMP_Lock (</b><br><b>OPAMP_HandleTypeDef * hopamp)</b> |
| Function Description | Lock the selected opamp configuration.                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hopamp</b> : OPAMP handle</li> </ul>  |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                         |

### 27.2.13 HAL\_OPAMP\_GetState

|                      |                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_OPAMP_StateTypeDef HAL_OPAMP_GetState (</b><br><b>OPAMP_HandleTypeDef * hopamp)</b> |
| Function Description | Return the OPAMP state.                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hopamp</b> : OPAMP handle</li> </ul>           |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                  |

## 27.3 OPAMP Firmware driver defines

### 27.3.1 OPAMP

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

- **IS\_OPAMP\_FACTORYTRIMMING**

***OPAMP InvertingInput***

- **OPAMP\_INVERTINGINPUT\_VM0**  
Comparator inverting input connected to dedicated IO pin low-leakage
- **OPAMP\_INVERTINGINPUT\_VM1**  
Comparator inverting input connected to alternative IO pin available on some device packages
- **OPAMP\_INVERTINGINPUT\_VINM**  
Alternate name for comparator inverting input connected to alternative IO pin available on some device packages
- **IOPAMP\_INVERTINGINPUT\_VM0**
- **IOPAMP\_INVERTINGINPUT\_VM1**
- **IS\_OPAMP\_INVERTING\_INPUT**

***OPAMP Mode***

- **OPAMP\_STANDALONE\_MODE**  
OPAMP standalone mode
- **OPAMP\_FOLLOWER\_MODE**  
OPAMP follower mode
- **IS\_OPAMP\_FUNCTIONAL\_NORMALMODE**

***OPAMP NonInvertingInput***

- **OPAMP\_NONINVERTINGINPUT\_VP0**  
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
- **OPAMP\_NONINVERTINGINPUT\_DAC\_CH2**  
Comparator non-inverting input connected internally to DAC channel 2. Available on OPAMP2 only.
- **IS\_OPAMP\_NONINVERTING\_INPUT**

***OPAMP PowerMode***

- **OPAMP\_POWERMODE\_NORMAL**
- **OPAMP\_POWERMODE\_LOWPOWER**
- **IS\_OPAMP\_POWERMODE**

***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)
- **IS\_OPAMP\_POWER\_SUPPLY\_RANGE**

***OPAMP Private Constants***

- **OPAMP\_TRIMMING\_DELAY**

***OPAMP Private Macro***

- **\_\_HAL\_OPAMP\_RESET\_HANDLE\_STATE**  
**Description:** Reset OPAMP handle state.  
**Parameters:** \_\_HANDLE\_\_: OPAMP handle.  
**Return value:**None:
- **\_\_OPAMP\_CSR\_OPAXPD**  
**Description:** Select the OPAMP bit OPAxPD (power-down) corresponding to the

- selected OPAMP instance.
- Parameters:** `__HANDLE__`: OPAMP handle
- Return value:**None
- **`__OPAMP_CSR_S3SELX`**  
**Description:** Select the OPAMP bit S3SELx (switch 3) corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_S4SELX`**  
**Description:** Select the OPAMP bit S4SELx (switch 4) corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_S5SELX`**  
**Description:** Select the OPAMP bit S5SELx (switch 5) corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_S6SELX`**  
**Description:** Select the OPAMP bit S6SELx (switch 6) corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_OPAXCAL_L`**  
**Description:** Select the OPAMP bit OPAxCAL\_L (offset calibration for differential pair P) corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_OPAXCAL_H`**  
**Description:** Select the OPAMP bit OPAxCAL\_H (offset calibration for differential pair N) corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_OPAXLPM`**  
**Description:** Select the OPAMP bit OPAxLPM (low power mode) corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_ALL_SWITCHES`**  
**Description:** Select the OPAMP bits of all switches corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_ANAWSELX`**  
**Description:** Select the OPAMP bit ANAWSELx (switch SanA) corresponding to the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_CSR_OPAXCALOUT`**  
**Description:** Select the OPAMP bit OPAxCALOUT in function of the selected OPAMP instance.  
**Parameters:** `__HANDLE__`: OPAMP handle  
**Return value:**None
  - **`__OPAMP_OFFSET_TRIM_BITPOSITION`**  
**Description:** Select the OPAMP trimming bits position value (position of LSB) in

register OPAMP\_OTR or register OPAMP\_LPOTR in function of the selected OPAMP instance and the transistors differential pair high (PMOS) or low (NMOS).

**Parameters:** \_\_HANDLE\_\_: OPAMP handle \_\_TRIM\_HIGH\_LOW\_\_: transistors differential pair high or low. Must be a value of

**Return value:**None:

- **\_\_OPAMP\_OFFSET\_TRIM\_SET**

**Description:** Shift the OPAMP trimming bits to register OPAMP\_OTR or register OPAMP\_LPOTR in function of the selected OPAMP instance and the transistors differential pair high (PMOS) or low (NMOS).

**Parameters:** \_\_HANDLE\_\_: OPAMP handle \_\_TRIM\_HIGH\_LOW\_\_: transistors differential pair high or low. Must be a value of \_\_TRIMMING\_VALUE\_\_: Trimming value

**Return value:**None:

- **IS\_OPAMP\_TRIMMINGVALUE**

**Description:** Check that trimming value is within correct range.

**Parameters:** TRIMMINGVALUE: OPAMP trimming value

**Return value:**None:

#### ***OPAMP UserTrimming***

- **OPAMP\_TRIMMING\_FACTORY**  
Factory trimming
- **OPAMP\_TRIMMING\_USER**  
User trimming
- **IS\_OPAMP\_TRIMMING**

## 28 HAL OPAMP Extension Driver

### 28.1 OPAMPEx Firmware driver API description

The following section lists the various functions of the OPAMPEx library.

#### 28.1.1 Peripheral Control functions

- OPAMP unlock.
- [\*\*HAL\\_OPAMPEx\\_Unlock\(\)\*\*](#)

#### 28.1.2 [\*\*HAL\\_OPAMPEx\\_SelfCalibrateAll\*\*](#)

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_OPAMPEx_SelfCalibrateAll (</b><br><b><i>OPAMP_HandleTypeDef</i> * hopamp1, <i>OPAMP_HandleTypeDef</i></b><br><b>* hopamp2, <i>OPAMP_HandleTypeDef</i> * hopamp3)</b>                                                                                                                                                                                                                                                                                                                                 |
| Function Description | Run the self calibration of the 3 OPAMPs in parallel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hopamp1</b> : handle</li> <li>• <b>hopamp2</b> : handle</li> <li>• <b>hopamp3</b> : handle</li> </ul>                                                                                                                                                                                                                                                                                                                                                                             |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>• Trimming values (PMOS &amp; NMOS) are updated and user trimming is enabled is calibration is successful.</li> <li>• 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.</li> <li>• Calibration runs about 10 ms (5 dichotomy steps, repeated for P and N transistors: 10 steps with 1 ms for each step).</li> </ul> |

#### 28.1.3 [\*\*HAL\\_OPAMPEx\\_Unlock\*\*](#)

|                      |                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_OPAMPEx_Unlock (</b><br><b><i>OPAMP_HandleTypeDef</i> * hopamp)</b> |
| Function Description | Unlock the selected opamp configuration.                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hopamp</b> : OPAMP handle</li> </ul>             |

- |               |                     |
|---------------|---------------------|
| Return values | • <b>HAL status</b> |
| Notes         | • None.             |

## 28.2 OPAMPEx Firmware driver defines

### 28.2.1 OPAMPEx

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**  
Return value:None:
- **\_\_HAL\_OPAMP\_OPAMP3OUT\_DISCONNECT\_ADC\_COMP1**  
Return value:None:

#### *OPAMPEx Private Macro*

- **\_\_OPAMP\_INSTANCE\_DECIMAL\_\_**  
**Description:** Get the OPAMP instance in decimal number for further processing needs by HAL OPAMP driver functions.  
**Parameters:** **\_\_HANDLE\_\_**: OPAMP handle  
**Return value:**0: for OPAMP1, "1" for OPAMP2, "2" for OPAMP3

## 29 HAL PCD Generic Driver

### 29.1 PCD Firmware driver registers structures

#### 29.1.1 PCD\_InitTypeDef

*PCD\_InitTypeDef* is defined in the `stm32l1xx_hal_pcd.h`

##### Data Fields

- `uint32_t dev_endpoints`
- `uint32_t speed`
- `uint32_t ep0_mps`
- `uint32_t phy_iface`
- `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_iface` 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.
- `uint32_t PCD_InitTypeDef::low_power_enable` Enable or disable Low Power mode
- `uint32_t PCD_InitTypeDef::lpm_enable` Enable or disable Battery charging.
- `uint32_t PCD_InitTypeDef::battery_charging_enable` Enable or disable Battery charging.

#### 29.1.2 PCD\_EPTTypeDef

*PCD\_EPTTypeDef* is defined in the `stm32l1xx_hal_pcd.h`

##### 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

### 29.1.3 PCD\_HandleTypeDef

*PCD\_HandleTypeDef* is defined in the *stm32l1xx\_hal\_pcd.h*

#### 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

## 29.2 PCD Firmware driver API description

The following section lists the various functions of the PCD library.

### 29.2.1 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
    - `__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();`

### 29.2.2 Initialization and de-initialization functions

This section provides functions allowing to:

- `HAL_PCD_Init()`
- `HAL_PCD_DelInit()`
- `HAL_PCD_MspInit()`
- `HAL_PCD_MspDelInit()`

### 29.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the PCD data transfers.

- `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_ISOOOUTIncompleteCallback()`

- `HAL_PCD_ISOINIncompleteCallback()`
- `HAL_PCD_ConnectCallback()`
- `HAL_PCD_DisconnectCallback()`

#### 29.2.4 Peripheral Control functions

This subsection provides a set of functions allowing to control the PCD data transfers.

- `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_ActiveRemoteWakeup()`
- `HAL_PCD_DeActiveRemoteWakeup()`

#### 29.2.5 Peripheral State functions

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

- `HAL_PCD_GetState()`
- `HAL_PCDEx_SetConnectionState()`

#### 29.2.6 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"> <li>• <code>hpcd</code> : PCD handle</li> </ul>                                 |
| Return values        | <ul style="list-style-type: none"> <li>• <code>HAL status</code></li> </ul>                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                          |

#### 29.2.7 HAL\_PCD\_DelInit

|                      |                                                                                    |
|----------------------|------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_PCD_DelInit ( <i>PCD_HandleTypeDef</i> * <i>hpcd</i>)</b> |
| Function Description | DeInitializes the PCD peripheral.                                                  |
| Parameters           | <ul style="list-style-type: none"><li>• <b><i>hpcd</i></b> : PCD handle</li></ul>  |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                            |

## 29.2.8 HAL\_PCD\_MspInit

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_MspInit ( <i>PCD_HandleTypeDef</i> * <i>hpcd</i>)</b>             |
| Function Description | Initializes the PCD MSP.                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b><i>hpcd</i></b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                           |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                           |

## 29.2.9 HAL\_PCD\_MspDelInit

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_MspDelInit ( <i>PCD_HandleTypeDef</i> * <i>hpcd</i>)</b>          |
| Function Description | DeInitializes PCD MSP.                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b><i>hpcd</i></b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                           |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                           |

## 29.2.10 HAL\_PCD\_Start

|               |                                                                                  |
|---------------|----------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_PCD_Start ( <i>PCD_HandleTypeDef</i> * <i>hpcd</i>)</b> |
|---------------|----------------------------------------------------------------------------------|

---

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Description | Start The USB OTG Device.                                                  |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.11 HAL\_PCD\_Stop

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_PCD_Stop ( <i>PCD_HandleTypeDef</i> * hpcd)</b>   |
| Function Description | Stop The USB OTG Device.                                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.12 HAL\_PCD\_IRQHandler

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_IRQHandler ( <i>PCD_HandleTypeDef</i> * hpcd)</b>          |
| Function Description | This function handles PCD interrupt request.                               |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.13 HAL\_PCD\_DataOutStageCallback

|                      |                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_DataOutStageCallback ( <i>PCD_HandleTypeDef</i> * hpcd, uint8_t epnum)</b> |
| Function Description | Data out stage callbacks.                                                                  |

|               |                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>epnum</b> : endpoint number</li></ul> |
| Return values | <ul style="list-style-type: none"><li>• None.</li></ul>                                                             |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                                                             |

### 29.2.14 HAL\_PCD\_DataInStageCallback

|                      |                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_DataInStageCallback ( <i>PCD_HandleTypeDef</i> * hpcd, uint8_t epnum)</b>                           |
| Function Description | Data IN stage callbacks.                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>epnum</b> : endpoint number</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                             |

### 29.2.15 HAL\_PCD\_SetupStageCallback

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_SetupStageCallback ( <i>PCD_HandleTypeDef</i> * hpcd)</b>  |
| Function Description | Setup stage callback.                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : pcd handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.16 HAL\_PCD\_SOFCallback

|               |                                                                    |
|---------------|--------------------------------------------------------------------|
| Function Name | <b>void HAL_PCD_SOFCallback ( <i>PCD_HandleTypeDef</i> * hpcd)</b> |
|---------------|--------------------------------------------------------------------|

---

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Description | USB Start Of Frame callbacks.                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.17 HAL\_PCD\_ResetCallback

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_ResetCallback ( <i>PCD_HandleTypeDef</i> * hpcd)</b>       |
| Function Description | USB Reset callbacks.                                                       |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.18 HAL\_PCD\_SuspendCallback

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_SuspendCallback ( <i>PCD_HandleTypeDef</i> * hpcd)</b>     |
| Function Description | Suspend event callbacks.                                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.19 HAL\_PCD\_ResumeCallback

|                      |                                                                       |
|----------------------|-----------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_ResumeCallback ( <i>PCD_HandleTypeDef</i> * hpcd)</b> |
| Function Description | Resume event callbacks.                                               |

|               |                                                                            |
|---------------|----------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

## 29.2.20 HAL\_PCD\_ISOOUTIncompleteCallback

|                      |                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_ISOOUTIncompleteCallback ( <i>PCD_HandleTypeDef</i> * hpcd, uint8_t epcnum)</b>                      |
| Function Description | Incomplete ISO OUT callbacks.                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>epcnum</b> : endpoint number</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                              |

## 29.2.21 HAL\_PCD\_ISOINIncompleteCallback

|                      |                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_ISOINIncompleteCallback ( <i>PCD_HandleTypeDef</i> * hpcd, uint8_t epcnum)</b>                       |
| Function Description | Incomplete ISO IN callbacks.                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>epcnum</b> : endpoint number</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                              |

## 29.2.22 HAL\_PCD\_ConnectCallback

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| Function Name | <b>void HAL_PCD_ConnectCallback ( <i>PCD_HandleTypeDef</i> * hpcd)</b> |
|---------------|------------------------------------------------------------------------|

---

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Description | Connection event callbacks.                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.23 HAL\_PCD\_DisconnectCallback

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCD_DisconnectCallback ( <i>PCD_HandleTypeDef</i> * hpcd)</b>  |
| Function Description | Disconnection event callbacks.                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : pcd handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

### 29.2.24 HAL\_PCD\_DevConnect

|                      |                                                                                |
|----------------------|--------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_PCD_DevConnect ( <i>PCD_HandleTypeDef</i> * hpcd)</b> |
| Function Description | Connect the USB device.                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul>     |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>            |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                        |

### 29.2.25 HAL\_PCD\_DevDisconnect

|               |                                                                                   |
|---------------|-----------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_PCD_DevDisconnect ( <i>PCD_HandleTypeDef</i> * hpcd)</b> |
|---------------|-----------------------------------------------------------------------------------|

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Description | Disconnect the USB device.                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

## 29.2.26 HAL\_PCD\_SetAddress

|                      |                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_PCD_SetAddress (</b><br><b><i>PCD_HandleTypeDef</i> * hpcd, uint8_t address)</b>                |
| Function Description | Set the USB Device address.                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>address</b> : new device address</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                  |

## 29.2.27 HAL\_PCD\_EP\_Open

|                      |                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_PCD_EP_Open (</b><br><b><i>PCD_HandleTypeDef</i> * hpcd, uint8_t ep_addr, uint16_t</b><br><b>ep_mps, uint8_t ep_type)</b>                                                                 |
| Function Description | Open and configure an endpoint.                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>ep_addr</b> : endpoint address</li><li>• <b>ep_mps</b> : endpoint max packet size</li><li>• <b>ep_type</b> : endpoint type</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                            |

### 29.2.28 HAL\_PCD\_EP\_Close

|                      |                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_PCD_EP_Close (<br/>    <i>PCD_HandleTypeDef</i> * hpcd, uint8_t ep_addr)</code>               |
| Function Description | Deactivate an endpoint.                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hpcd</b> : PCD handle</li> <li>• <b>ep_addr</b> : endpoint address</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                 |

### 29.2.29 HAL\_PCD\_EP\_Receive

|                      |                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_PCD_EP_Receive (<br/>    <i>PCD_HandleTypeDef</i> * hpcd, uint8_t ep_addr, uint8_t * pBuf,<br/>    uint32_t len)</code>                                                                                      |
| Function Description | Receive an amount of data.                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hpcd</b> : PCD handle</li> <li>• <b>ep_addr</b> : endpoint address</li> <li>• <b>pBuf</b> : pointer to the reception buffer</li> <li>• <b>len</b> : amount of data to be received</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                |

### 29.2.30 HAL\_PCD\_EP\_GetRxCount

|                      |                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint16_t HAL_PCD_EP_GetRxCount (<i>PCD_HandleTypeDef</i> * hpcd, uint8_t ep_addr)</code>                            |
| Function Description | Get Received Data Size.                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hpcd</b> : PCD handle</li> <li>• <b>ep_addr</b> : endpoint address</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Data Size</b></li> </ul>                                                      |

## Notes

- None.

### 29.2.31 HAL\_PCD\_EP\_Transmit

|                      |                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_PCD_EP_Transmit (<br/>    <i>PCD_HandleTypeDef</i> * hpcd, uint8_t ep_addr, uint8_t * pBuf,<br/>    uint32_t len)</code>                                                                               |
| Function Description | Send an amount of data.                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>ep_addr</b> : endpoint address</li><li>• <b>pBuf</b> : pointer to the transmission buffer</li><li>• <b>len</b> : amount of data to be sent</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                            |

### 29.2.32 HAL\_PCD\_EP\_SetStall

|                      |                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_PCD_EP_SetStall (<br/>    <i>PCD_HandleTypeDef</i> * hpcd, uint8_t ep_addr)</code>         |
| Function Description | Set a STALL condition over an endpoint.                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>ep_addr</b> : endpoint address</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                    |

### 29.2.33 HAL\_PCD\_EP\_ClrStall

|               |                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_PCD_EP_ClrStall (<br/>    <i>PCD_HandleTypeDef</i> * hpcd, uint8_t ep_addr)</code> |
|---------------|----------------------------------------------------------------------------------------------------------------|

---

|                      |                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------|
| Function Description | Clear a STALL condition over in an endpoint.                                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>ep_addr</b> : endpoint address</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                |

### 29.2.34 HAL\_PCD\_EP\_Flush

|                      |                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_PCD_EP_Flush (</b><br><b><i>PCD_HandleTypeDef</i> * hpcd, uint8_t ep_addr)</b>                |
| Function Description | Flush an endpoint.                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li><li>• <b>ep_addr</b> : endpoint address</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                |

### 29.2.35 HAL\_PCD\_ActiveRemoteWakeup

|                      |                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_PCD_ActiveRemoteWakeup (</b><br><b><i>PCD_HandleTypeDef</i> * hpcd)</b> |
| Function Description | HAL_PCD_ActiveRemoteWakeup : active remote wakeup signalling.                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hpcd</b> : PCD handle</li></ul>                       |
| Return values        | <ul style="list-style-type: none"><li>• <b>status</b></li></ul>                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                          |

### 29.2.36 HAL\_PCD\_DeActiveRemoteWakeup

---

|                      |                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_PCD_DeActiveRemoteWakeup ( <i>PCD_HandleTypeDef</i> * <i>hpcd</i>)</b> |
| Function Description | HAL_PCD_DeActiveRemoteWakeup : de-active remote wakeup signalling.                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hpcd</b> : PCD handle</li> </ul>                    |
| Return values        | <ul style="list-style-type: none"> <li>• <b>status</b></li> </ul>                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                       |

### 29.2.37 HAL\_PCD\_GetState

|                      |                                                                                    |
|----------------------|------------------------------------------------------------------------------------|
| Function Name        | <b>PCD_StateTypeDef HAL_PCD_GetState ( <i>PCD_HandleTypeDef</i> * <i>hpcd</i>)</b> |
| Function Description | Return the PCD state.                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hpcd</b> : PCD handle</li> </ul>       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                          |

### 29.2.38 HAL\_PCDEx\_SetConnectionState

|                      |                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PCDEx_SetConnectionState ( <i>PCD_HandleTypeDef</i> * <i>hpcd</i>, <i>uint8_t</i> <i>state</i>)</b>     |
| Function Description | Software Device Connection.                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hpcd</b> : PCD handle</li> <li>• <b>state</b> : Device state</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                           |

## 29.3 PCD Firmware driver defines

### 29.3.1 PCD

PCD

*PCD Core PHY*

- PCD\_PHY\_EMBEDDED

*PCD Core Speed*

- PCD\_SPEED\_HIGH
- PCD\_SPEED\_FULL

*PCD-ENDP-Type*

- PCD\_ENDP0
- PCD\_ENDP1
- PCD\_ENDP2
- PCD\_ENDP3
- PCD\_ENDP4
- PCD\_ENDP5
- PCD\_ENDP6
- PCD\_ENDP7
- PCD\_SNG\_BUF
- PCD\_DBL\_BUF
- IS\_PCD\_ALL\_INSTANCE

*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\_EXTI\_ENABLE\_IT
- \_\_HAL\_USB\_EXTI\_DISABLE\_IT
- \_\_HAL\_USB\_EXTI\_GET\_FLAG
- \_\_HAL\_USB\_EXTI\_CLEAR\_FLAG
- \_\_HAL\_USB\_EXTI\_SET\_RISING\_EDGE\_TRIGGER
- \_\_HAL\_USB\_EXTI\_SET\_FALLING\_EDGE\_TRIGGER
- \_\_HAL\_USB\_EXTI\_SET\_FALLINGRISING\_TRIGGER

*PCD\_Exti\_Line\_Wakeup*

- **USB\_EXTI\_LINE\_WAKEUP**

External interrupt line 18 Connected to the USB FS EXTI Line

**PCD Private Constants**

- **BTABLE\_ADDRESS**

**PCD Private Macros**

- **PCD\_SET\_ENDPOINT**

- **PCD\_GET\_ENDPOINT**

- **PCD\_SET\_EPTYPE**

**Description:** sets the type in the endpoint register(bits EP\_TYPE[1:0])

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number. wType: Endpoint Type.

**Return value:**None

- **PCD\_GET\_EPTYPE**

**Description:** gets the type in the endpoint register(bits EP\_TYPE[1:0])

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.

**Return value:**Endpoint Type

- **PCD\_FreeUserBuffer**

**Description:** free buffer used from the application realizing it to the line toggles bit SW\_BUF in the double buffered endpoint register

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number. bDir: Direction

**Return value:**None

- **PCD\_GET\_DB\_DIR**

**Description:** gets direction of the double buffered endpoint

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.

**Return value:**EP\_DBUF\_OUT: if the endpoint counter not yet programmed.

- **PCD\_SET\_EP\_TX\_STATUS**

**Description:** sets the status for tx transfer (bits STAT\_TX[1:0]).

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number. wState: new state

**Return value:**None

- **PCD\_SET\_EP\_RX\_STATUS**

**Description:** sets the status for rx transfer (bits STAT\_RX[1:0])

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number. wState: new state

**Return value:**None

- **PCD\_SET\_EP\_TXRX\_STATUS**

**Description:** sets the status for rx & tx (bits STAT\_TX[1:0] & STAT\_RX[1:0])

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number. wStaterx: new state. wStatetx: new state.

**Return value:**None

- **PCD\_GET\_EP\_TX\_STATUS**

**Description:** gets the status for tx/rx transfer (bits STAT\_TX[1:0] /STAT\_RX[1:0])

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.

**Return value:**status

- **PCD\_GET\_EP\_RX\_STATUS**

- **PCD\_SET\_EP\_TX\_VALID**

**Description:** sets directly the VALID tx/rx-status into the endpoint register

**Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint

- Number.
- Return value:**None:
- **PCD\_SET\_EP\_RX\_VALID**
  - **PCD\_GET\_EP\_TX\_STALL\_STATUS**
- Description:** checks stall condition in an endpoint.
- Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.
- Return value:**TRUE: = endpoint in stall condition.
- **PCD\_GET\_EP\_RX\_STALL\_STATUS**
  - **PCD\_SET\_EP\_KIND**
- Description:** set & clear EP\_KIND bit.
- Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.
- Return value:**None:
- **PCD\_CLEAR\_EP\_KIND**
  - **PCD\_SET\_OUT\_STATUS**
- Description:** Sets/clears directly STATUS\_OUT bit in the endpoint register.
- Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.
- Return value:**None:
- **PCD\_CLEAR\_OUT\_STATUS**
  - **PCD\_SET\_EP\_DBUF**
- Description:** Sets/clears directly EP\_KIND bit in the endpoint register.
- Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.
- Return value:**None:
- **PCD\_CLEAR\_EP\_DBUF**
  - **PCD\_CLEAR\_RX\_EP\_CTR**
- Description:** Clears bit CTR\_RX / CTR\_TX in the endpoint register.
- Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.
- Return value:**None:
- **PCD\_CLEAR\_TX\_EP\_CTR**
  - **PCD\_RX\_DTOG**
- Description:** Toggles DTOG\_RX / DTOG\_TX bit in the endpoint register.
- Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.
- Return value:**None:
- **PCD\_TX\_DTOG**
  - **PCD\_CLEAR\_RX\_DTOG**
- Description:** Clears DTOG\_RX / DTOG\_TX bit in the endpoint register.
- Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number.
- Return value:**None:
- **PCD\_CLEAR\_TX\_DTOG**
  - **PCD\_SET\_EP\_ADDRESS**
- Description:** Sets address in an endpoint register.
- Parameters:**USBbx: USB peripheral instance register address. bEpNum: Endpoint Number. bAddr: Address.
- Return value:**None:
- **PCD\_GET\_EP\_ADDRESS**
  - **PCD\_EP\_TX\_ADDRESS**
  - **PCD\_EP\_TX\_CNT**
  - **PCD\_EP\_RX\_ADDRESS**

- **PCD\_EP\_RX\_CNT**
- **PCD\_SET\_EP\_RX\_CNT**
- **PCD\_SET\_EP\_TX\_ADDRESS**  
**Description:** sets address of the tx/rx buffer.  
**Parameters:** USBx: USB peripheral instance register address. bEpNum: Endpoint Number. wAddr: address to be set (must be word aligned).  
**Return value:**None:
- **PCD\_SET\_EP\_RX\_ADDRESS**
- **PCD\_GET\_EP\_TX\_ADDRESS**  
**Description:** Gets address of the tx/rx buffer.  
**Parameters:** USBx: USB peripheral instance register address. bEpNum: Endpoint Number.  
**Return value:**address: of the buffer.
- **PCD\_GET\_EP\_RX\_ADDRESS**
- **PCD\_CALC\_BLK32**  
**Description:** Sets counter of rx buffer with no.  
**Parameters:** dwReg: Register wCount: Counter. wNBlocks: no. of Blocks.  
**Return value:**None:
- **PCD\_CALC\_BLK2**
- **PCD\_SET\_EP\_CNT\_RX\_REG**
- **PCD\_SET\_EP\_RX\_DBUF0\_CNT**
- **PCD\_SET\_EP\_TX\_CNT**  
**Description:** sets counter for the tx/rx buffer.  
**Parameters:** USBx: USB peripheral instance register address. bEpNum: Endpoint Number. wCount: Counter value.  
**Return value:**None:
- **PCD\_GET\_EP\_TX\_CNT**  
**Description:** gets counter of the tx buffer.  
**Parameters:** USBx: USB peripheral instance register address. bEpNum: Endpoint Number.  
**Return value:**Counter: value
- **PCD\_GET\_EP\_RX\_CNT**
- **PCD\_SET\_EP\_DBUF0\_ADDR**  
**Description:** Sets buffer 0/1 address in a double buffer endpoint.  
**Parameters:** USBx: USB peripheral instance register address. bEpNum: Endpoint Number. wBuf0Addr: buffer 0 address.  
**Return value:**Counter: value
- **PCD\_SET\_EP\_DBUF1\_ADDR**
- **PCD\_SET\_EP\_DBUF\_ADDR**  
**Description:** Sets addresses in a double buffer endpoint.  
**Parameters:** USBx: USB peripheral instance register address. bEpNum: Endpoint Number. wBuf0Addr: buffer 0 address. wBuf1Addr: = buffer 1 address.  
**Return value:**None:
- **PCD\_GET\_EP\_DBUF0\_ADDR**  
**Description:** Gets buffer 0/1 address of a double buffer endpoint.  
**Parameters:** USBx: USB peripheral instance register address. bEpNum: Endpoint Number.  
**Return value:**None:
- **PCD\_GET\_EP\_DBUF1\_ADDR**
- **PCD\_SET\_EP\_DBUF0\_CNT**  
**Description:** Gets buffer 0/1 address of a double buffer endpoint.  
**Parameters:** USBx: USB peripheral instance register address. bEpNum: Endpoint Number. bDir: endpoint dir EP\_DBUF\_OUT = OUT EP\_DBUF\_IN = IN wCount:

Counter value

**Return value:**None:

- **PCD\_SET\_EP\_DBUF1\_CNT**
- **PCD\_SET\_EP\_DBUF\_CNT**
- **PCD\_GET\_EP\_DBUF0\_CNT**

**Description:** Gets buffer 0/1 rx/tx counter for double buffering.

**Parameters:**USBx: USB peripheral instance register address. bEpNum: Endpoint Number.

**Return value:**None:

- **PCD\_GET\_EP\_DBUF1\_CNT**

## 30 HAL PCD Extension Driver

### 30.1 PCDEEx Firmware driver API description

The following section lists the various functions of the PCDEEx library.

#### 30.1.1 Peripheral Control functions

This section provides functions allowing to:

- Configure PMA for the EndPoint
- [\*HAL\\_PCDEx\\_PMACConfig\(\)\*](#)

#### 30.1.2 [\*HAL\\_PCDEx\\_PMACConfig\*](#)

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_PCDEx_PMACConfig (</code><br><a href="#"><i>PCD_HandleTypeDef</i></a> * <code>hpcd</code> , <code>uint16_t ep_addr</code> , <code>uint16_t ep_kind</code> , <code>uint32_t pmaaddress</code> )                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Function Description | Configure PMA for EP.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b><code>hpcd</code></b> : Device instance</li><li>• <b><code>ep_addr</code></b> : endpoint address</li><li>• <b><code>ep_kind</code></b> : endpoint Kind USB_SNG_BUF: Single Buffer used<br/>USB_DBL_BUF: Double Buffer used</li><li>• <b><code>pmaaddress</code></b> : 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.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>: status</b></li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

### 30.2 PCDEEx Firmware driver defines

#### 30.2.1 PCDEEx

PCDEEx

## 31 HAL PWR Generic Driver

### 31.1 PWR Firmware driver registers structures

#### 31.1.1 PWR\_PVDTTypeDef

*PWR\_PVDTTypeDef* is defined in the `stm32l1xx_hal_pwr.h`

##### Data Fields

- *uint32\_t PVDLevel*
- *uint32\_t Mode*

##### Field Documentation

- *uint32\_t PWR\_PVDTTypeDef::PVDLevel*: Specifies the PVD detection level. This parameter can be a value of [\*PWR\\_PVD\\_detection\\_level\*](#)
- *uint32\_t PWR\_PVDTTypeDef::Mode*: Specifies the operating mode for the selected pins. This parameter can be a value of [\*PWR\\_PVD\\_Mode\*](#)

### 31.2 PWR Firmware driver API description

The following section lists the various functions of the PWR library.

#### 31.2.1 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 `__PWR_CLK_ENABLE()` macro.
- Enable access to RTC domain using the `HAL_PWR_EnableBkUpAccess()` function.
- [`HAL\_PWR\_DeInit\(\)`](#)
- [`HAL\_PWR\_EnableBkUpAccess\(\)`](#)
- [`HAL\_PWR\_DisableBkUpAccess\(\)`](#)

#### 31.2.2 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\_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.
- `HAL_PWR_PVDConfig()`
- `HAL_PWR_EnablePVD()`
- `HAL_PWR_DisablePVD()`
- `HAL_PWR_EnableWakeUpPin()`
- `HAL_PWR_DisableWakeUpPin()`
- `HAL_PWR_EnterSLEEPMode()`
- `HAL_PWR_EnterSTOPMode()`
- `HAL_PWR_EnterSTANDBYMode()`
- `HAL_PWR_PVD_IRQHandler()`
- `HAL_PWR_PVDCALLBACK()`

### 31.2.3 HAL\_PWR\_DeInit

|                      |                                                                                                                                                                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_DeInit ( void )</b>                                                                                                                                                                                                                                                                                                  |
| Function Description | Deinitializes the PWR peripheral registers to their default reset values.                                                                                                                                                                                                                                                            |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> <li>• ULP and FWU bits are not reset by this function.</li> </ul> |

### 31.2.4 HAL\_PWR\_EnableBkUpAccess

|                      |                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_EnableBkUpAccess ( void )</b>                                                                                                                      |
| Function Description | Enables access to the backup domain (RTC registers, RTC backup data registers ).                                                                                   |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• If the HSE divided by 2, 4, 8 or 16 is used as the RTC clock, the Backup Domain Access should be kept enabled.</li> </ul> |

### 31.2.5 HAL\_PWR\_DisableBkUpAccess

---

|                      |                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_DisableBkUpAccess ( void )</b>                                                                                                                   |
| Function Description | Disables access to the backup domain (RTC registers, RTC backup data registers).                                                                                 |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>If the HSE divided by 2, 4, 8 or 16 is used as the RTC clock, the Backup Domain Access should be kept enabled.</li> </ul> |

### 31.2.6 HAL\_PWR\_PVDConfig

|                      |                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_PVDConfig ( <i>PWR_PVDTTypeDef</i> *<br/>sConfigPVD)</b>                                                                                                                               |
| Function Description | Configures the voltage threshold detected by the Power Voltage Detector(PVD).                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li><b>sConfigPVD</b> : pointer to an <i>PWR_PVDTTypeDef</i> structure that contains the configuration information for the PVD.</li> </ul>                          |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>Refer to the electrical characteristics of your device datasheet for more details about the voltage threshold corresponding to each detection level.</li> </ul> |

### 31.2.7 HAL\_PWR\_EnablePVD

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_EnablePVD ( void )</b>                  |
| Function Description | Enables the Power Voltage Detector(PVD).                |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul> |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul> |

### 31.2.8 HAL\_PWR\_DisablePVD

---

|                      |                                                       |
|----------------------|-------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_DisablePVD ( void )</b>               |
| Function Description | Disables the Power Voltage Detector(PVD).             |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul> |

### 31.2.9 HAL\_PWR\_EnableWakeUpPin

|                      |                                                                                                                                                                                                                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_EnableWakeUpPin ( uint32_t WakeUpPinx )</b>                                                                                                                                                                                                                                                                                |
| Function Description | Enables the WakeUp PINx functionality.                                                                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li><b>WakeUpPinx</b> : Specifies the Power Wake-Up pin to enable. This parameter can be one of the following values:<ul style="list-style-type: none"><li><b>PWR_WAKEUP_PIN1</b></li><li><b>PWR_WAKEUP_PIN2</b></li><li><b>PWR_WAKEUP_PIN3</b> Only on product with GPIOE available</li></ul></li></ul> |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                                                                                      |

### 31.2.10 HAL\_PWR\_DisableWakeUpPin

|                      |                                                                                                                                                                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_DisableWakeUpPin ( uint32_t WakeUpPinx )</b>                                                                                                                                                                                                                                                                                |
| Function Description | Disables the WakeUp PINx functionality.                                                                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li><b>WakeUpPinx</b> : Specifies the Power Wake-Up pin to disable. This parameter can be one of the following values:<ul style="list-style-type: none"><li><b>PWR_WAKEUP_PIN1</b></li><li><b>PWR_WAKEUP_PIN2</b></li><li><b>PWR_WAKEUP_PIN3</b> Only on product with GPIOE available</li></ul></li></ul> |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                                                                                       |

### 31.2.11 HAL\_PWR\_EnterSLEEPMode

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_PWR_EnterSLEEPMode ( uint32_t Regulator,<br/>uint8_t SLEEPEntry)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Function Description | Enters Sleep mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Regulator</b> : Specifies the regulator state in SLEEP mode. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>PWR_MAINREGULATOR_ON</b> SLEEP mode with regulator ON</li> <li>– <b>PWR_LOWPOWERREGULATOR_ON</b> SLEEP mode with low power regulator ON</li> </ul> </li> <li>• <b>SLEEPEntry</b> : 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"> <li>– <b>PWR_SLEEPENTRY_WFI</b> enter SLEEP mode with WFI instruction</li> <li>– <b>PWR_SLEEPENTRY_WFE</b> enter SLEEP mode with WFE instruction</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• In Sleep mode, all I/O pins keep the same state as in Run mode.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

### 31.2.12 HAL\_PWR\_EnterSTOPMode

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_PWR_EnterSTOPMode ( uint32_t Regulator, uint8_t STOPEntry)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Function Description | Enters Stop mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Regulator</b> : Specifies the regulator state in Stop mode. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>PWR_MAINREGULATOR_ON</b> Stop mode with regulator ON</li> <li>– <b>PWR_LOWPOWERREGULATOR_ON</b> Stop mode with low power regulator ON</li> </ul> </li> <li>• <b>STOPEntry</b> : 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"> <li>– <b>PWR_STOPENTRY_WFI</b> Enter Stop mode with WFI instruction</li> <li>– <b>PWR_STOPENTRY_WFE</b> Enter Stop mode with WFE instruction</li> </ul> </li> </ul> |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Notes         | <ul style="list-style-type: none"><li>In Stop mode, all I/O pins keep the same state as in Run mode.</li><li>When exiting Stop mode by using an interrupt or a wakeup event, MSI RC oscillator is selected as system clock.</li><li>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.</li></ul> |

### 31.2.13 HAL\_PWR\_EnterSTANDBYMode

|                      |                                                                                                                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_EnterSTANDBYMode ( void )</b>                                                                                                                                                                                                                                                                                      |
| Function Description | Enters Standby mode.                                                                                                                                                                                                                                                                                                               |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>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.</li></ul> |

### 31.2.14 HAL\_PWR\_PVD\_IRQHandler

|                      |                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_PVD_IRQHandler ( void )</b>                                                           |
| Function Description | This function handles the PWR PVD interrupt request.                                                  |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                 |
| Notes                | <ul style="list-style-type: none"><li>This API should be called under the PVD_IRQHandler().</li></ul> |

### 31.2.15 HAL\_PWR\_PVDCallback

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| Function Name        | <b>void HAL_PWR_PVDCallback ( void )</b>                  |
| Function Description | PWR PVD interrupt callback.                               |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul> |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul> |

## 31.3 PWR Firmware driver defines

### 31.3.1 PWR

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 Macro*

- \_\_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

- **PWR EXTI LINE\_PVD**

External interrupt line 16 Connected to the PVD EXTI Line

- **\_\_HAL\_PWR\_PVD\_EXTI\_ENABLE\_IT**

**Return value:** None..

- **\_\_HAL\_PWR\_PVD\_EXTI\_DISABLE\_IT**

**Return value:** None..

- **\_\_HAL\_PWR\_PVD\_EXTI\_ENABLE\_EVENT**

**Return value:** None..

- **\_\_HAL\_PWR\_PVD\_EXTI\_DISABLE\_EVENT**

**Return value:** None..

- **\_\_HAL\_PWR\_PVD\_EXTI\_CLEAR\_EDGE\_TRIGGER**

**Return value:** None..

- **\_\_HAL\_PWR\_PVD\_EXTI\_SET\_FALLING\_EDGE\_TRIGGER**

**Return value:** None..

- **\_\_HAL\_PWR\_PVD\_EXTI\_SET\_RISING\_EDGE\_TRIGGER**

**Return value:** None..

- **\_\_HAL\_PWR\_PVD\_EXTI\_GET\_FLAG**

**Return value:** EXTI: PVD Line Status.

- **\_\_HAL\_PWR\_PVD\_EXTI\_CLEAR\_FLAG**

**Return value:** None..

- **\_\_HAL\_PWR\_PVD\_EXTI\_GENERATE\_SWIT**

**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\_PVDEVEL\_0**
- **PWR\_PVDEVEL\_1**
- **PWR\_PVDEVEL\_2**

- **PWR\_PVDLEVEL\_3**
- **PWR\_PVDLEVEL\_4**
- **PWR\_PVDLEVEL\_5**
- **PWR\_PVDLEVEL\_6**
- **PWR\_PVDLEVEL\_7**
- **IS\_PWR\_PVD\_LEVEL**

***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
- **IS\_PWR\_PVD\_MODE**

***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**
- **IS\_PWR\_REGULATOR**

***PWR Regulator Voltage Scale***

- **PWR\_REGULATOR\_VOLTAGE\_SCALE1**
- **PWR\_REGULATOR\_VOLTAGE\_SCALE2**
- **PWR\_REGULATOR\_VOLTAGE\_SCALE3**
- **IS\_PWR\_VOLTAGE\_SCALING\_RANGE**

***PWR SLEEP mode entry***

- **PWR\_SLEEPENTRY\_WFI**
- **PWR\_SLEEPENTRY\_WFE**
- **IS\_PWR\_SLEEP\_ENTRY**

***PWR STOP mode entry***

- **PWR\_STOPENTRY\_WFI**
- **PWR\_STOPENTRY\_WFE**
- **IS\_PWR\_STOP\_ENTRY**

## 32 HAL PWR Extension Driver

### 32.1 PWREx Firmware driver API description

The following section lists the various functions of the PWREx library.

#### 32.1.1 Peripheral extended features functions

- [\*HAL\\_PWREx\\_EnableFastWakeUp\(\)\*](#)
- [\*HAL\\_PWREx\\_DisableFastWakeUp\(\)\*](#)
- [\*HAL\\_PWREx\\_EnableUltraLowPower\(\)\*](#)
- [\*HAL\\_PWREx\\_DisableUltraLowPower\(\)\*](#)
- [\*HAL\\_PWREx\\_EnableLowPowerRunMode\(\)\*](#)
- [\*HAL\\_PWREx\\_DisableLowPowerRunMode\(\)\*](#)

#### 32.1.2 [\*\*HAL\\_PWREx\\_EnableFastWakeUp\*\*](#)

|                      |                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWREx_EnableFastWakeUp ( void )</b>                                                                                                                                                  |
| Function Description | Enables the Fast WakeUp from Ultra Low Power mode.                                                                                                                                               |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"><li>• 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.</li></ul> |

#### 32.1.3 [\*\*HAL\\_PWREx\\_DisableFastWakeUp\*\*](#)

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_PWREx_DisableFastWakeUp ( void )</b>        |
| Function Description | Disables the Fast WakeUp from Ultra Low Power mode.     |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

#### 32.1.4 [\*\*HAL\\_PWREx\\_EnableUltraLowPower\*\*](#)

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_PWREx_EnableUltraLowPower ( void )</b>      |
| Function Description | Enables the Ultra Low Power mode.                       |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 32.1.5 HAL\_PWREx\_DisableUltraLowPower

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_PWREx_DisableUltraLowPower ( void )</b>     |
| Function Description | Disables the Ultra Low Power mode.                      |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

### 32.1.6 HAL\_PWREx\_EnableLowPowerRunMode

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_PWREx_EnableLowPowerRunMode ( void )</b>                                                                                                                                                                                                                                                                                                                                                                      |
| Function Description | Enters the Low Power Run mode.                                                                                                                                                                                                                                                                                                                                                                                            |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• 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.</li><li>• In Low power run mode, all I/O pins keep the same state as in Run mode.</li></ul> |

### 32.1.7 HAL\_PWREx\_DisableLowPowerRunMode

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| Function Name        | <b>void HAL_PWREx_DisableLowPowerRunMode ( void )</b>   |
| Function Description | Exits the Low Power Run mode.                           |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul> |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul> |

## 32.2 PWREx Firmware driver defines

### 32.2.1 PWREx

PWREx

#### *PWREx Wakeup Pins*

- **PWR\_WAKEUP\_PIN1**
- **PWR\_WAKEUP\_PIN2**
- **PWR\_WAKEUP\_PIN3**
- **IS\_PWR\_WAKEUP\_PIN**

## 33 HAL RCC Generic Driver

### 33.1 RCC Firmware driver registers structures

#### 33.1.1 RCC\_PLLInitTypeDef

*RCC\_PLLInitTypeDef* is defined in the `stm32l1xx_hal_rcc.h`

##### Data Fields

- `uint32_t PLLState`
- `uint32_t PLLSource`
- `uint32_t PLLMUL`
- `uint32_t PLLDIV`

##### Field Documentation

- `uint32_t RCC_PLLInitTypeDef::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](#)

#### 33.1.2 RCC\_OscInitTypeDef

*RCC\_OscInitTypeDef* is defined in the `stm32l1xx_hal_rcc.h`

##### 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_OsclInitTypeDef::HSIState`** 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 = 0x1F
- **`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 = 0xFF
- **`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

### 33.1.3 **RCC\_ClkInitTypeDef**

**RCC\_ClkInitTypeDef** is defined in the `stm32l1xx_hal_rcc.h`

#### Data Fields

- **`uint32_t ClockType`**
- **`uint32_t SYSSCLKSource`**
- **`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::SYSSCLKSource`** The clock source (SYSSCLK) 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 (SYSSCLK). 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](#)

## 33.2 **RCC Firmware driver API description**

The following section lists the various functions of the RCC library.

### 33.2.1 **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) busses; all peripherals mapped on these busses 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 busses 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

### 33.2.2 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 busses 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 two 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 clock is 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 busses 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 busses. You can use "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 \_\_HAL\_RCC\_RTC\_CONFIG() and \_\_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 \_\_HAL\_RCC\_LCD\_CONFIG() macros to configure this clock. USB OTG FS and RTC: 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 (see table below).
3. The following table gives the different clock source frequencies depending on the product voltage range (see table below).

**Table 21: Number of wait states (WS) according to CPU clock (HCLK) frequency**

| <b>Latency</b>   | <b>HCLK clock frequency (MHz)</b> |                                |                                |
|------------------|-----------------------------------|--------------------------------|--------------------------------|
|                  | <b>voltage range 1 (1.8 V)</b>    | <b>voltage range 2 (1.5 V)</b> | <b>voltage range 3 (1.2 V)</b> |
| 0W(1CPU cycles)  | 0 < HCLK ≤ 16                     | 0 < HCLK ≤ 8                   | 0 < HCLK ≤ 2                   |
| 1WS(2CPU cycles) | 16 < HCLK ≤ 32                    | 8 < HCLK ≤ 16                  | 2 < HCLK ≤ 4                   |

**Table 22: Clock frequency versus product voltage range**

| <b>Product voltage range</b> | <b>Clock frequency</b> |            |                                                 |                              |
|------------------------------|------------------------|------------|-------------------------------------------------|------------------------------|
|                              | <b>MSI</b>             | <b>HSI</b> | <b>HSE</b>                                      | <b>PLL</b>                   |
| Range 1 (1.8 V)              | 4.2 MHz                | 16 MHz     | 32 MHz HSE (external clock) or 24 MHz (crystal) | 32 MHz (PLLVCO max = 96 MHz) |
| Range 2 (1.5 V)              | 4.2 MHz                | 16 MHz     | 16 MHz                                          | 16 MHz (PLLVCO max = 48 MHz) |
| Range 3 (1.2 V)              | 4.2 MHz                | NA         | 8 MHz                                           | 4 MHz (PLLVCO max = 24 MHz)  |

- [\*\*HAL\\_RCC\\_DelInit\(\)\*\*](#)
- [\*\*HAL\\_RCC\\_OscConfig\(\)\*\*](#)
- [\*\*HAL\\_RCC\\_ClockConfig\(\)\*\*](#)

### 33.2.3 Peripheral Control functions

This subsection provides a set of functions allowing to control the RCC Clocks frequencies.

- [\*\*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_CCSCallback()`

### 33.2.4 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"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• 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</li> <li>• This function doesn't modify the configuration of the Peripheral clocksLSI, LSE and RTC clocks</li> </ul> |

### 33.2.5 HAL\_RCC\_OscConfig

|                      |                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RCC_OscConfig (</code><br><code>RCC_OscInitTypeDef * RCC_OscInitStruct)</code>                                                                                     |
| Function Description | Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef.                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>RCC_OscInitStruct</b> : pointer to an RCC_OscInitTypeDef structure that contains the configuration information for the RCC Oscillators.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• The PLL is not disabled when used as system clock.</li> </ul>                                                                                         |

### 33.2.6 HAL\_RCC\_ClockConfig

|                      |                                                                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RCC_ClockConfig (</code><br><code>RCC_ClkInitTypeDef * RCC_ClkInitStruct, uint32_t FLatency)</code> |
| Function Description | Initializes the CPU, AHB and APB busses clocks according to the specified parameters in the RCC_ClkInitStruct.                  |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li><b>RCC_ClkInitStruct</b> : pointer to an RCC_OsclInitTypeDef structure that contains the configuration information for the RCC peripheral.</li> <li><b>FLatency</b> : FLASH Latency This parameter can be one of the following values:           <ul style="list-style-type: none"> <li><b>FLASH_LATENCY_0</b> FLASH Zero Latency cycle</li> <li><b>FLASH_LATENCY_1</b> FLASH One Latency cycle</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Return values | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Notes         | <ul style="list-style-type: none"> <li>The SystemCoreClock CMSIS variable is used to store System Clock Frequency and updated by HAL_RCC_GetHCLKFreq() function called within this function</li> <li>The MSI is used (enabled by hardware) as system clock source after startup 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).</li> <li>A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup 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.</li> <li>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")</li> </ul> |

### 33.2.7 HAL\_RCC\_MCOConfig

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_RCC_MCOConfig ( uint32_t RCC_MCOx, uint32_t RCC_MCOsource, uint32_t RCC_MCODiv)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Function Description | Selects the clock source to output on MCO pin.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li><b>RCC_MCOx</b> : specifies the output direction for the clock source. This parameter can be one of the following values:           <ul style="list-style-type: none"> <li><b>RCC_MCO</b> Clock source to output on MCO1 pin(PA8).</li> </ul> </li> <li><b>RCC_MCOsource</b> : specifies the clock source to output. This parameter can be one of the following values:           <ul style="list-style-type: none"> <li><b>RCC_MCO1SOURCE_NOCLOCK</b> No clock selected</li> <li><b>RCC_MCO1SOURCE_SYSCLK</b> System clock selected</li> <li><b>RCC_MCO1SOURCE_HSI</b> HSI oscillator clock selected</li> <li><b>RCC_MCO1SOURCE_MSISI</b> MSI oscillator clock selected</li> <li><b>RCC_MCO1SOURCE_HSE</b> HSE oscillator clock selected</li> <li><b>RCC_MCO1SOURCE_PLLCLK</b> PLL clock selected</li> <li><b>RCC_MCO1SOURCE_LSI</b> LSI clock selected</li> </ul> </li> </ul> |

- **RCC\_MCO1SOURCE\_LSE** LSE clock selected
  - **RCC\_MCODiv** : specifies the MCO DIV. This parameter can be one of the following values:
    - **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
- None.
- Notes
- MCO pin should be configured in alternate function mode.

### 33.2.8 HAL\_RCC\_EnableCSS

|                      |                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RCC_EnableCSS ( void )</b>                                                                                                                                                                                                                                                                                                                                                                |
| Function Description | Enables the Clock Security System.                                                                                                                                                                                                                                                                                                                                                                    |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> </ul> |

### 33.2.9 HAL\_RCC\_DisableCSS

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| Function Name        | <b>void HAL_RCC_DisableCSS ( void )</b>                   |
| Function Description | Disables the Clock Security System.                       |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul> |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul> |

### 33.2.10 HAL\_RCC\_GetSysClockFreq

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_RCC_GetSysClockFreq ( void )</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Function Description | Returns the SYSCLK frequency.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Return values        | <ul style="list-style-type: none"> <li>• <b>SYSCLK frequency</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• 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:</li> <li>• If SYSCLK source is MSI, function returns values based on MSI Value as defined by the MSI range.</li> <li>• If SYSCLK source is HSI, function returns values based on HSI_VALUE(*)</li> <li>• If SYSCLK source is HSE, function returns values based on HSE_VALUE(**)</li> <li>• If SYSCLK source is PLL, function returns values based on HSE_VALUE(**) or HSI_VALUE(*) multiplied/divided by the PLL factors.</li> <li>• (*) HSI_VALUE is a constant defined in <code>stm32l1xx_hal_conf.h</code> file (default value 16 MHz) but the real value may vary depending on the variations in voltage and temperature.</li> <li>• (**) HSE_VALUE is a constant defined in <code>stm32l1xx_hal_conf.h</code> 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.</li> <li>• The result of this function could be not correct when using fractional value for HSE crystal.</li> <li>• This function can be used by the user application to compute the baudrate for the communication peripherals or configure other parameters.</li> <li>• 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.</li> </ul> |

### 33.2.11 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"> <li>• <b>HCLK frequency</b></li> </ul>                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• 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.</li> <li>• The SystemCoreClock CMSIS variable is used to store</li> </ul> |

---

System Clock Frequency and updated within this function

### 33.2.12 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"><li><b>PCLK1 frequency</b></li></ul>                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"><li>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.</li></ul> |

### 33.2.13 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"><li><b>PCLK2 frequency</b></li></ul>                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"><li>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.</li></ul> |

### 33.2.14 HAL\_RCC\_GetOscConfig

|                      |                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_RCC_GetOscConfig ( <i>RCC_OscInitTypeDef</i> * RCC_OscInitStruct)</code>                                                        |
| Function Description | Configures the RCC_OscInitStruct according to the internal RCC configuration registers.                                                        |
| Parameters           | <ul style="list-style-type: none"><li><b>RCC_OscInitStruct</b> : pointer to an RCC_OscInitTypeDef structure that will be configured.</li></ul> |

---

|               |                                                       |
|---------------|-------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>None.</li></ul> |
| Notes         | <ul style="list-style-type: none"><li>None.</li></ul> |

### 33.2.15 HAL\_RCC\_GetClockConfig

|                      |                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RCC_GetClockConfig ( <i>RCC_ClkInitTypeDef</i> *<br/>RCC_ClkInitStruct, uint32_t * pFLatency)</b>                                                                                           |
| Function Description | Configures the RCC_ClkInitStruct according to the internal RCC configuration registers.                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li><b>RCC_ClkInitStruct</b> : pointer to an RCC_ClkInitTypeDef structure that will be configured.</li><li><b>pFLatency</b> : Pointer on the Flash Latency.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                   |

### 33.2.16 HAL\_RCC\_NMI\_IRQHandler

|                      |                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RCC_NMI_IRQHandler ( void )</b>                                                        |
| Function Description | This function handles the RCC CSS interrupt request.                                               |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                                              |
| Notes                | <ul style="list-style-type: none"><li>This API should be called under the NMI_Handler().</li></ul> |

### 33.2.17 HAL\_RCC\_CCSCallback

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| Function Name        | <b>void HAL_RCC_CCSCallback ( void )</b>                    |
| Function Description | RCC Clock Security System interrupt callback.               |
| Return values        | <ul style="list-style-type: none"><li><b>none</b></li></ul> |

Notes

- None.

## 33.3 RCC Firmware driver defines

### 33.3.1 RCC

RCC

*RCC AHB Clock Source*

- RCC\_SYSCLK\_DIV1
- RCC\_SYSCLK\_DIV2
- RCC\_SYSCLK\_DIV4
- RCC\_SYSCLK\_DIV8
- RCC\_SYSCLK\_DIV16
- RCC\_SYSCLK\_DIV64
- RCC\_SYSCLK\_DIV128
- RCC\_SYSCLK\_DIV256
- RCC\_SYSCLK\_DIV512

*RCC APB1 APB2 Clock Source*

- RCC\_HCLK\_DIV1
- RCC\_HCLK\_DIV2
- RCC\_HCLK\_DIV4
- RCC\_HCLK\_DIV8
- RCC\_HCLK\_DIV16

*RCC BitAddress AliasRegion*

- RCC\_OFFSET
- RCC\_CR\_OFFSET
- RCC\_CFGR\_OFFSET
- RCC\_CIR\_OFFSET
- RCC\_CSR\_OFFSET
- RCC\_CR\_OFFSET\_BB
- RCC\_CFGR\_OFFSET\_BB
- RCC\_CIR\_OFFSET\_BB
- RCC\_CSR\_OFFSET\_BB
- HSION\_BITNUMBER
- CR\_HSION\_BB
- MSION\_BITNUMBER
- CR\_MSION\_BB
- HSEON\_BITNUMBER
- CR\_HSEON\_BB
- CSSON\_BITNUMBER
- CR\_CSSON\_BB
- PLLON\_BITNUMBER
- CR\_PLLON\_BB
- LSION\_BITNUMBER
- CSR\_LSION\_BB
- LSEON\_BITNUMBER



- **CSR\_LSEON\_BB**
- **LSEBYP\_BITNUMBER**
- **CSR\_LSEBYP\_BB**
- **RTCEN\_BITNUMBER**
- **CSR\_RTCEN\_BB**
- **RTCRST\_BITNUMBER**
- **CSR\_RTCRST\_BB**
- **CR\_BYTE2\_ADDRESS**
- **CIR\_BYTE1\_ADDRESS**
- **CIR\_BYTE2\_ADDRESS**

#### **RCC Exported Macros**

- **\_HAL\_RCC\_HSI\_ENABLE**
- **\_HAL\_RCC\_HSI\_DISABLE**
- **\_HAL\_RCC\_HSE\_CONFIG**

**Description:** Macros to enable or disable the External High Speed oscillator (HSE).

**Parameters:** **\_HSE\_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

- **\_HAL\_RCC\_MSI\_ENABLE**
- **\_HAL\_RCC\_MSI\_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.

- **\_HAL\_RCC\_MSI\_CALIBRATIONVALUE\_ADJUST**

**Description:** macro to adjust the 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 0x1F.

- **\_HAL\_RCC\_MSI\_RANGE\_CONFIG**
- **\_HAL\_RCC\_LSI\_ENABLE**
- **\_HAL\_RCC\_LSI\_DISABLE**
- **\_HAL\_RCC\_LSE\_CONFIG**
- **\_HAL\_RCC\_RTC\_ENABLE**
- **\_HAL\_RCC\_RTC\_DISABLE**
- **\_HAL\_RCC\_BACKUPRESET\_FORCE**
- **\_HAL\_RCC\_BACKUPRESET\_RELEASE**
- **\_HAL\_RCC\_RTC\_CLKPRESCALER**

**Description:** Macro to configures the RTC clock (RTCCLK).

**Parameters:** **\_RTC\_CLKSOURCE\_**: specifies the RTC clock source. This parameter can be one of the following values: 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\_DIV16: HSE divided by 16 selected as RTC clock

- **\_HAL\_RCC\_RTC\_CONFIG**
- **\_HAL\_RCC\_GET\_RTC\_SOURCE**
- **\_HAL\_RCC\_PLL\_ENABLE**

- **`_HAL_RCC_PLL_DISABLE`**
- **`_HAL_RCC_PLL_CONFIG`**

**Description:** macros 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 `_PLLMUL_`: 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 `_PLLDIV_`: specifies the division factor for PLL VCO input clock This parameter can be one of the following values: `RCC_PLL_DIV2`: PLL clock output =  $\text{PLL}\bar{\text{VCO}} / 2$  `RCC_PLL_DIV3`: PLL clock output =  $\text{PLL}\bar{\text{VCO}} / 3$  `RCC_PLL_DIV4`: PLL clock output =  $\text{PLL}\bar{\text{VCO}} / 4$

- **`_HAL_RCC_GET_SYSCLK_SOURCE`**

**Return value:** The clock source used as system clock. The returned value can be one of the following: `RCC_CFGR_SWS_MSI`: MSI used as system clock `RCC_CFGR_SWS_HSI`: HSI used as system clock `RCC_CFGR_SWS_HSE`: HSE used as system clock `RCC_CFGR_SWS_PLL`: PLL used as system clock

- **`_HAL_RCC_ENABLE_IT`**

**Description:** macros to manage the specified RCC Flags and interrupts.

**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 device)

- **`_HAL_RCC_DISABLE_IT`**

**Description:** Disable RCC interrupt (Perform Byte access to `RCC_CIR[14:8]` bits to disable the selected interrupts).

**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 device)

- **`_HAL_RCC_CLEAR_IT`**

**Description:** Clear the RCC's interrupt pending bits ( Perform Byte access to `RCC_CIR[23:16]` bits to clear the selected 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 device)  
`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 device) RCC\_IT\_CSS: Clock Security System interrupt

**Return value:** The new state of `_INTERRUPT_` (TRUE or FALSE).

- `_HAL_RCC_CLEAR_RESET_FLAGS`
- `_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\_BORRST: POR/PDR or BOR 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\_LPWRST: Low Power reset.

**Return value:** The new state of `_FLAG_` (TRUE or FALSE).

- `_HAL_RCC_GET_PLL_OSCSOURCE`

**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

#### **RCC Flag**

- `CR_REG_INDEX`
- `CSR_REG_INDEX`
- `RCC_FLAG_HSIRDY`
- `RCC_FLAG_MSIRDY`
- `RCC_FLAG_HSERDY`
- `RCC_FLAG_PLLRDY`
- `RCC_FLAG_LSIRDY`
- `RCC_FLAG_LSERDY`
- `RCC_FLAG_LSECSS`
- `RCC_FLAG_RMV`
- `RCC_FLAG_OBLRST`
- `RCC_FLAG_PINRST`
- `RCC_FLAG_PORRST`
- `RCC_FLAG_SFTRST`
- `RCC_FLAG_IWDGRST`
- `RCC_FLAG_WWDGRST`
- `RCC_FLAG_LPWRST`
- `RCC_FLAG_MASK`

#### **RCC HSE Config**

- `RCC_HSE_OFF`
- `RCC_HSE_ON`
- `RCC_HSE_BYPASS`

#### **RCC HSI Config**

- `RCC_HSI_OFF`
- `RCC_HSI_ON`
- `RCC_HSICALIBRATION_DEFAULT`

*RCC Interrupt*

- `RCC_IT_LSIRDY`
- `RCC_IT_LSERDY`
- `RCC_IT_HSIRDY`
- `RCC_IT_HSERDY`
- `RCC_IT_PLLRDY`
- `RCC_IT_MSIRDY`
- `RCC_IT_LSECSS`
- `RCC_IT_CSS`

*RCC LSE Config*

- `RCC_LSE_OFF`
- `RCC_LSE_ON`
- `RCC_LSE_BYPASS`

*RCC LSI Config*

- `RCC_LSI_OFF`
- `RCC_LSI_ON`

*RCC 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`

*RCC MCO1 Clock Prescaler*

- `RCC_MCODIV_1`
- `RCC_MCODIV_2`
- `RCC_MCODIV_4`
- `RCC_MCODIV_8`
- `RCC_MCODIV_16`

*RCC MCO Index*

- `RCC_MCO1`
- `RCC_MCO`

*RCC 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

#### *RCC MSI Config*

- **RCC\_MSI\_OFF**
- **RCC\_MSI\_ON**
- **RCC\_MSICALIBRATION\_DEFAULT**

#### *RCC Oscillator Type*

- **RCC\_OSCILLATORTYPE\_NONE**
- **RCC\_OSCILLATORTYPE\_HSE**
- **RCC\_OSCILLATORTYPE\_HSI**
- **RCC\_OSCILLATORTYPE\_LSE**
- **RCC\_OSCILLATORTYPE\_LSI**
- **RCC\_OSCILLATORTYPE\_MSI**

#### *RCC Peripheral Clock Enable Disable*

- **\_GPIOA\_CLK\_ENABLE**
- **\_GPIOB\_CLK\_ENABLE**
- **\_GPIOC\_CLK\_ENABLE**
- **\_GPIOD\_CLK\_ENABLE**
- **\_GPIOH\_CLK\_ENABLE**
- **\_CRC\_CLK\_ENABLE**
- **\_FLITF\_CLK\_ENABLE**
- **\_DMA1\_CLK\_ENABLE**
- **\_GPIOA\_CLK\_DISABLE**
- **\_GPIOB\_CLK\_DISABLE**
- **\_GPIOC\_CLK\_DISABLE**
- **\_GPIOD\_CLK\_DISABLE**
- **\_GPIOH\_CLK\_DISABLE**
- **\_CRC\_CLK\_DISABLE**
- **\_FLITF\_CLK\_DISABLE**
- **\_DMA1\_CLK\_DISABLE**
- **\_TIM2\_CLK\_ENABLE**
- **\_TIM3\_CLK\_ENABLE**
- **\_TIM4\_CLK\_ENABLE**
- **\_TIM6\_CLK\_ENABLE**
- **\_TIM7\_CLK\_ENABLE**
- **\_WWDG\_CLK\_ENABLE**
- **\_SPI2\_CLK\_ENABLE**
- **\_USART2\_CLK\_ENABLE**
- **\_USART3\_CLK\_ENABLE**
- **\_I2C1\_CLK\_ENABLE**
- **\_I2C2\_CLK\_ENABLE**
- **\_USB\_CLK\_ENABLE**
- **\_PWR\_CLK\_ENABLE**
- **\_DAC\_CLK\_ENABLE**
- **\_COMP\_CLK\_ENABLE**
- **\_TIM2\_CLK\_DISABLE**
- **\_TIM3\_CLK\_DISABLE**
- **\_TIM4\_CLK\_DISABLE**
- **\_TIM6\_CLK\_DISABLE**
- **\_TIM7\_CLK\_DISABLE**

- \_\_WWDG\_CLK\_DISABLE
- \_\_SPI2\_CLK\_DISABLE
- \_\_USART2\_CLK\_DISABLE
- \_\_USART3\_CLK\_DISABLE
- \_\_I2C1\_CLK\_DISABLE
- \_\_I2C2\_CLK\_DISABLE
- \_\_USB\_CLK\_DISABLE
- \_\_PWR\_CLK\_DISABLE
- \_\_DAC\_CLK\_DISABLE
- \_\_COMP\_CLK\_DISABLE
- \_\_SYSCFG\_CLK\_ENABLE
- \_\_TIM9\_CLK\_ENABLE
- \_\_TIM10\_CLK\_ENABLE
- \_\_TIM11\_CLK\_ENABLE
- \_\_ADC1\_CLK\_ENABLE
- \_\_SPI1\_CLK\_ENABLE
- \_\_USART1\_CLK\_ENABLE
- \_\_SYSCFG\_CLK\_DISABLE
- \_\_TIM9\_CLK\_DISABLE
- \_\_TIM10\_CLK\_DISABLE
- \_\_TIM11\_CLK\_DISABLE
- \_\_ADC1\_CLK\_DISABLE
- \_\_SPI1\_CLK\_DISABLE
- \_\_USART1\_CLK\_DISABLE

#### *RCC Peripheral Clock Force Release*

- \_\_AHB\_FORCE\_RESET
- \_\_GPIOA\_FORCE\_RESET
- \_\_GPIOB\_FORCE\_RESET
- \_\_GPIOC\_FORCE\_RESET
- \_\_GPIOD\_FORCE\_RESET
- \_\_GPIOH\_FORCE\_RESET
- \_\_CRC\_FORCE\_RESET
- \_\_FLITF\_FORCE\_RESET
- \_\_DMA1\_FORCE\_RESET
- \_\_AHB\_RELEASE\_RESET
- \_\_GPIOA\_RELEASE\_RESET
- \_\_GPIOB\_RELEASE\_RESET
- \_\_GPIOC\_RELEASE\_RESET
- \_\_GPIOD\_RELEASE\_RESET
- \_\_GPIOH\_RELEASE\_RESET
- \_\_CRC\_RELEASE\_RESET
- \_\_FLITF\_RELEASE\_RESET
- \_\_DMA1\_RELEASE\_RESET
- \_\_APB1\_FORCE\_RESET
- \_\_TIM2\_FORCE\_RESET
- \_\_TIM3\_FORCE\_RESET
- \_\_TIM4\_FORCE\_RESET
- \_\_TIM6\_FORCE\_RESET
- \_\_TIM7\_FORCE\_RESET
- \_\_WWDG\_FORCE\_RESET
- \_\_SPI2\_FORCE\_RESET
- \_\_USART2\_FORCE\_RESET

- \_\_USART3\_FORCE\_RESET
- \_\_I2C1\_FORCE\_RESET
- \_\_I2C2\_FORCE\_RESET
- \_\_USB\_FORCE\_RESET
- \_\_PWR\_FORCE\_RESET
- \_\_DAC\_FORCE\_RESET
- \_\_COMP\_FORCE\_RESET
- \_\_APB1\_RELEASE\_RESET
- \_\_TIM2\_RELEASE\_RESET
- \_\_TIM3\_RELEASE\_RESET
- \_\_TIM4\_RELEASE\_RESET
- \_\_TIM6\_RELEASE\_RESET
- \_\_TIM7\_RELEASE\_RESET
- \_\_WWDG\_RELEASE\_RESET
- \_\_SPI2\_RELEASE\_RESET
- \_\_USART2\_RELEASE\_RESET
- \_\_USART3\_RELEASE\_RESET
- \_\_I2C1\_RELEASE\_RESET
- \_\_I2C2\_RELEASE\_RESET
- \_\_USB\_RELEASE\_RESET
- \_\_PWR\_RELEASE\_RESET
- \_\_DAC\_RELEASE\_RESET
- \_\_COMP\_RELEASE\_RESET
- \_\_APB2\_FORCE\_RESET
- \_\_SYSCFG\_FORCE\_RESET
- \_\_TIM9\_FORCE\_RESET
- \_\_TIM10\_FORCE\_RESET
- \_\_TIM11\_FORCE\_RESET
- \_\_ADC1\_FORCE\_RESET
- \_\_SPI1\_FORCE\_RESET
- \_\_USART1\_FORCE\_RESET
- \_\_APB2\_RELEASE\_RESET
- \_\_SYSCFG\_RELEASE\_RESET
- \_\_TIM9\_RELEASE\_RESET
- \_\_TIM10\_RELEASE\_RESET
- \_\_TIM11\_RELEASE\_RESET
- \_\_ADC1\_RELEASE\_RESET
- \_\_SPI1\_RELEASE\_RESET
- \_\_USART1\_RELEASE\_RESET

#### RCC Peripheral Clock Sleep Enable Disable

- \_\_GPIOA\_CLK\_SLEEP\_ENABLE
- \_\_GPIOB\_CLK\_SLEEP\_ENABLE
- \_\_GPIOC\_CLK\_SLEEP\_ENABLE
- \_\_GPIOD\_CLK\_SLEEP\_ENABLE
- \_\_GPIOH\_CLK\_SLEEP\_ENABLE
- \_\_CRC\_CLK\_SLEEP\_ENABLE
- \_\_FLITF\_CLK\_SLEEP\_ENABLE
- \_\_DMA1\_CLK\_SLEEP\_ENABLE
- \_\_GPIOA\_CLK\_SLEEP\_DISABLE
- \_\_GPIOB\_CLK\_SLEEP\_DISABLE
- \_\_GPIOC\_CLK\_SLEEP\_DISABLE
- \_\_GPIOD\_CLK\_SLEEP\_DISABLE

- \_\_GPIOH\_CLK\_SLEEP\_DISABLE
- \_\_CRC\_CLK\_SLEEP\_DISABLE
- \_\_FLITF\_CLK\_SLEEP\_DISABLE
- \_\_DMA1\_CLK\_SLEEP\_DISABLE
- \_\_TIM2\_CLK\_SLEEP\_ENABLE
- \_\_TIM3\_CLK\_SLEEP\_ENABLE
- \_\_TIM4\_CLK\_SLEEP\_ENABLE
- \_\_TIM6\_CLK\_SLEEP\_ENABLE
- \_\_TIM7\_CLK\_SLEEP\_ENABLE
- \_\_WWDG\_CLK\_SLEEP\_ENABLE
- \_\_SPI2\_CLK\_SLEEP\_ENABLE
- \_\_USART2\_CLK\_SLEEP\_ENABLE
- \_\_USART3\_CLK\_SLEEP\_ENABLE
- \_\_I2C1\_CLK\_SLEEP\_ENABLE
- \_\_I2C2\_CLK\_SLEEP\_ENABLE
- \_\_USB\_CLK\_SLEEP\_ENABLE
- \_\_PWR\_CLK\_SLEEP\_ENABLE
- \_\_DAC\_CLK\_SLEEP\_ENABLE
- \_\_COMP\_CLK\_SLEEP\_ENABLE
- \_\_TIM2\_CLK\_SLEEP\_DISABLE
- \_\_TIM3\_CLK\_SLEEP\_DISABLE
- \_\_TIM4\_CLK\_SLEEP\_DISABLE
- \_\_TIM6\_CLK\_SLEEP\_DISABLE
- \_\_TIM7\_CLK\_SLEEP\_DISABLE
- \_\_WWDG\_CLK\_SLEEP\_DISABLE
- \_\_SPI2\_CLK\_SLEEP\_DISABLE
- \_\_USART2\_CLK\_SLEEP\_DISABLE
- \_\_USART3\_CLK\_SLEEP\_DISABLE
- \_\_I2C1\_CLK\_SLEEP\_DISABLE
- \_\_I2C2\_CLK\_SLEEP\_DISABLE
- \_\_USB\_CLK\_SLEEP\_DISABLE
- \_\_PWR\_CLK\_SLEEP\_DISABLE
- \_\_DAC\_CLK\_SLEEP\_DISABLE
- \_\_COMP\_CLK\_SLEEP\_DISABLE
- \_\_SYSCFG\_CLK\_SLEEP\_ENABLE
- \_\_TIM9\_CLK\_SLEEP\_ENABLE
- \_\_TIM10\_CLK\_SLEEP\_ENABLE
- \_\_TIM11\_CLK\_SLEEP\_ENABLE
- \_\_ADC1\_CLK\_SLEEP\_ENABLE
- \_\_SPI1\_CLK\_SLEEP\_ENABLE
- \_\_USART1\_CLK\_SLEEP\_ENABLE
- \_\_SYSCFG\_CLK\_SLEEP\_DISABLE
- \_\_TIM9\_CLK\_SLEEP\_DISABLE
- \_\_TIM10\_CLK\_SLEEP\_DISABLE
- \_\_TIM11\_CLK\_SLEEP\_DISABLE
- \_\_ADC1\_CLK\_SLEEP\_DISABLE
- \_\_SPI1\_CLK\_SLEEP\_DISABLE
- \_\_USART1\_CLK\_SLEEP\_DISABLE

#### RCC PLL Clock Source

- RCC\_PLLSOURCE\_HSI
- RCC\_PLLSOURCE\_HSE

***RCC PLL Config***

- `RCC_PLL_NONE`
- `RCC_PLL_OFF`
- `RCC_PLL_ON`

***RCC PLL Division Factor***

- `RCC_PLL_DIV2`
- `RCC_PLL_DIV3`
- `RCC_PLL_DIV4`

***RCC 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`

***RCC Private Defines***

- `HSE_TIMEOUT_VALUE`
- `MSI_TIMEOUT_VALUE`
- `HSI_TIMEOUT_VALUE`
- `LSI_TIMEOUT_VALUE`
- `PLL_TIMEOUT_VALUE`
- `CLOCKSWITCH_TIMEOUT_VALUE`
- `DBP_TIMEOUT_VALUE`
- `LSE_TIMEOUT_VALUE`

***RCC Private Macros***

- `_MCO1_CLK_ENABLE`
- `MCO1_GPIO_PORT`
- `MCO1_PIN`
- `IS_RCC_PLLSOURCE`
- `IS_RCC_OSCILLATORTYPE`
- `IS_RCC_HSE`
- `IS_RCC_LSE`
- `IS_RCC_HSI`
- `IS_RCC_CALIBRATION_VALUE`
- `IS_RCC_MSIRANGE`
- `IS_RCC_LSI`
- `IS_RCC_MSI`
- `IS_RCC_PLL`
- `IS_RCC_PLL_DIV`
- `IS_RCC_PLL_MUL`
- `IS_RCC_CLOCKTYPE`
- `IS_RCC_SYSCLKSOURCE`
- `IS_RCC_HCLK`
- `IS_RCC_PCLK`
- `IS_RCC_MCO`

- **IS\_RCC\_MCODIV**
- **IS\_RCC\_MCO1SOURCE**

***RCC RTC LCD Clock Source***

- **RCC\_RTCCLKSOURCE\_LSE**
- **RCC\_RTCCLKSOURCE\_LSI**
- **RCC\_RTCCLKSOURCE\_HSE\_DIV2**
- **RCC\_RTCCLKSOURCE\_HSE\_DIV4**
- **RCC\_RTCCLKSOURCE\_HSE\_DIV8**
- **RCC\_RTCCLKSOURCE\_HSE\_DIV16**

***RCC System Clock Source***

- **RCC\_SYSCLKSOURCE\_MSI**
- **RCC\_SYSCLKSOURCE\_HSI**
- **RCC\_SYSCLKSOURCE\_HSE**
- **RCC\_SYSCLKSOURCE\_PLLCLK**

***RCC System Clock Type***

- **RCC\_CLOCKTYPE\_SYSCLK**
- **RCC\_CLOCKTYPE\_HCLK**
- **RCC\_CLOCKTYPE\_PCLK1**
- **RCC\_CLOCKTYPE\_PCLK2**

## 34 HAL RCC Extension Driver

### 34.1 RCCEEx Firmware driver registers structures

#### 34.1.1 RCC\_PeriphCLKInitTypeDef

*RCC\_PeriphCLKInitTypeDef* is defined in the `stm32l1xx_hal_rcc_ex.h`

##### 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 [RCCEEx\\_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](#)

### 34.2 RCCEEx Firmware driver API description

The following section lists the various functions of the RCCEEx library.

#### 34.2.1 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) and `RCC_BDCR` register are set to their reset values.

- [HAL\\_RCCEEx\\_PeriphCLKConfig\(\)](#)
- [HAL\\_RCCEEx\\_GetPeriphCLKConfig\(\)](#)
- [HAL\\_RCCEEx\\_EnableLSECSS\(\)](#)
- [HAL\\_RCCEEx\\_DisableLSECSS\(\)](#)

#### 34.2.2 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 <b>RCC_PeriphCLKInitTypeDef</b> .                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>PeriphClkInit</b> : pointer to an <b>RCC_PeriphCLKInitTypeDef</b> structure that contains the configuration information for the Extended Peripherals clocks(RTC/LCD clock).</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                          |

**34.2.3 HAL\_RCCEEx\_GetPeriphCLKConfig**

|                      |                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RCCEEx_GetPeriphCLKConfig (</b><br><b><i>RCC_PeriphCLKInitTypeDef * PeriphClkInit)</i></b>                                                                                                                             |
| Function Description | Get the PeriphClkInit according to the internal RCC configuration registers.                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>PeriphClkInit</b> : pointer to an <b>RCC_PeriphCLKInitTypeDef</b> structure that returns the configuration information for the Extended Peripherals clocks(RTC/LCD clocks).</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                          |

**34.2.4 HAL\_RCCEEx\_EnableLSECSS**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RCCEEx_EnableLSECSS ( void )</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Function Description | Enables the LSE Clock Security System.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• 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</li> </ul> |

- action to secure the application.
- LSE CSS available only for high density and medium+ devices

### 34.2.5 HAL\_RCCEEx\_DisableLSECSS

|                      |                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RCCEEx_DisableLSECSS ( void )</b>                                                                                                                                                                                                                                                                                             |
| Function Description | Disables the LSE Clock Security System.                                                                                                                                                                                                                                                                                                   |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>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).</li> <li>LSE CSS available only for high density and medium+ devices</li> </ul> |

## 34.3 RCCEEx Firmware driver defines

### 34.3.1 RCCEEx

RCCEEx

#### *RCCEEx Exported Macros*

- \_\_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 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\_DIV16: HSE divided by 16 selected as RTC clock

- \_\_HAL\_RCC\_GET\_LCD\_SOURCE

#### *RCCEEx Force Release Peripheral Reset*

- \_\_GPIOE\_FORCE\_RESET
- \_\_GPIOE\_RELEASE\_RESET
- \_\_GPIOF\_FORCE\_RESET
- \_\_GPIOG\_FORCE\_RESET
- \_\_GPIOF\_RELEASE\_RESET
- \_\_GPIOG\_RELEASE\_RESET

- \_\_DMA2\_FORCE\_RESET
- \_\_DMA2\_RELEASE\_RESET
- \_\_CRYP\_FORCE\_RESET
- \_\_CRYP\_RELEASE\_RESET
- \_\_FSMC\_FORCE\_RESET
- \_\_FSMC\_RELEASE\_RESET
- \_\_LCD\_FORCE\_RESET
- \_\_LCD\_RELEASE\_RESET
- \_\_TIM5\_FORCE\_RESET
- \_\_TIM5\_RELEASE\_RESET
- \_\_SPI3\_FORCE\_RESET
- \_\_SPI3\_RELEASE\_RESET
- \_\_UART4\_FORCE\_RESET
- \_\_UART5\_FORCE\_RESET
- \_\_UART4\_RELEASE\_RESET
- \_\_UART5\_RELEASE\_RESET
- \_\_OPAMP\_FORCE\_RESET
- \_\_OPAMP\_RELEASE\_RESET
- \_\_SDIO\_FORCE\_RESET
- \_\_SDIO\_RELEASE\_RESET

*RCCEx\_Peripheral\_Clock\_Enable\_Disable*

- \_\_GPIOE\_CLK\_ENABLE
- \_\_GPIOE\_CLK\_DISABLE
- \_\_GPIOF\_CLK\_ENABLE
- \_\_GPIOG\_CLK\_ENABLE
- \_\_GPIOF\_CLK\_DISABLE
- \_\_GPIOG\_CLK\_DISABLE
- \_\_DMA2\_CLK\_ENABLE
- \_\_DMA2\_CLK\_DISABLE
- \_\_CRYP\_CLK\_ENABLE
- \_\_CRYP\_CLK\_DISABLE
- \_\_FSMC\_CLK\_ENABLE
- \_\_FSMC\_CLK\_DISABLE
- \_\_LCD\_CLK\_ENABLE
- \_\_LCD\_CLK\_DISABLE
- \_\_TIM5\_CLK\_ENABLE
- \_\_TIM5\_CLK\_DISABLE
- \_\_SPI3\_CLK\_ENABLE
- \_\_SPI3\_CLK\_DISABLE
- \_\_UART4\_CLK\_ENABLE
- \_\_UART5\_CLK\_ENABLE
- \_\_UART4\_CLK\_DISABLE
- \_\_UART5\_CLK\_DISABLE
- \_\_OPAMP\_CLK\_ENABLE
- \_\_OPAMP\_CLK\_DISABLE
- \_\_SDIO\_CLK\_ENABLE
- \_\_SDIO\_CLK\_DISABLE

*RCCEx\_Peripheral\_Clock\_Sleep\_Enable\_Disable*

- \_\_GPIOE\_CLK\_SLEEP\_ENABLE
- \_\_GPIOE\_CLK\_SLEEP\_DISABLE
- \_\_GPIOF\_CLK\_SLEEP\_ENABLE

- \_\_GPIOG\_CLK\_SLEEP\_ENABLE
- \_\_GPIOF\_CLK\_SLEEP\_DISABLE
- \_\_GPIOG\_CLK\_SLEEP\_DISABLE
- \_\_DMA2\_CLK\_SLEEP\_ENABLE
- \_\_DMA2\_CLK\_SLEEP\_DISABLE
- \_\_CRYP\_CLK\_SLEEP\_ENABLE
- \_\_CRYP\_CLK\_SLEEP\_DISABLE
- \_\_FSMC\_CLK\_SLEEP\_ENABLE
- \_\_FSMC\_CLK\_SLEEP\_DISABLE
- \_\_LCD\_CLK\_SLEEP\_ENABLE
- \_\_LCD\_CLK\_SLEEP\_DISABLE
- \_\_TIM5\_CLK\_SLEEP\_ENABLE
- \_\_TIM5\_CLK\_SLEEP\_DISABLE
- \_\_SPI3\_CLK\_SLEEP\_ENABLE
- \_\_SPI3\_CLK\_SLEEP\_DISABLE
- \_\_UART4\_CLK\_SLEEP\_ENABLE
- \_\_UART5\_CLK\_SLEEP\_ENABLE
- \_\_UART4\_CLK\_SLEEP\_DISABLE
- \_\_UART5\_CLK\_SLEEP\_DISABLE
- \_\_SDIO\_CLK\_SLEEP\_ENABLE
- \_\_SDIO\_CLK\_SLEEP\_DISABLE

*RCCEx Periph Clock Selection*

- RCC\_PERIPHCLK\_RTC
- RCC\_PERIPHCLK\_LCD

*Private Defines*

- LSECSSON\_BITNUMBER
- CSR\_LSECSSON\_BB

*Private Macros*

- IS\_RCC\_PERIPH\_CLOCK

## 35 HAL RTC Generic Driver

### 35.1 RTC Firmware driver registers structures

#### 35.1.1 RTC\_InitTypeDef

*RTC\_InitTypeDef* is defined in the `stm32l1xx_hal_rtc.h`

##### 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 = 0x7FFF
- *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](#)

#### 35.1.2 RTC\_DateTypeDef

*RTC\_DateTypeDef* is defined in the `stm32l1xx_hal_rtc.h`

##### 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

### 35.1.3 RTC\_HandleTypeDef

*RTC\_HandleTypeDef* is defined in the `stm32l1xx_hal_rtc.h`

#### 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

## 35.2 RTC Firmware driver API description

The following section lists the various functions of the RTC library.

### 35.2.1 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

### 35.2.2 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.

### 35.2.3 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 \_\_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.

### 35.2.4 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.

### 35.2.5 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.

### 35.2.6 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).
  - [\*HAL\\_RTC\\_Init\(\)\*](#)
  - [\*HAL\\_RTC\\_DelInit\(\)\*](#)
  - [\*HAL\\_RTC\\_MspInit\(\)\*](#)
  - [\*HAL\\_RTC\\_MspDelInit\(\)\*](#)
  - [\*HAL\\_RTC\\_SetTime\(\)\*](#)
  - [\*HAL\\_RTC\\_GetTime\(\)\*](#)
  - [\*HAL\\_RTC\\_SetDate\(\)\*](#)
  - [\*HAL\\_RTC\\_GetDate\(\)\*](#)

### 35.2.7 RTC Time and Date functions

This section provides functions allowing to configure Time and Date features

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

### 35.2.8 RTC Alarm functions

This section provides functions allowing to configure Alarm feature

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

### 35.2.9 Peripheral State functions

This subsection provides functions allowing to

- Get RTC state
- [\*HAL\\_RTC\\_GetState\(\)\*](#)

### 35.2.10 Peripheral Control functions

This subsection provides functions allowing to

- Wait for RTC Time and Date Synchronization
- [\*HAL\\_RTC\\_WaitForSynchro\(\)\*](#)

### 35.2.11 HAL\_RTC\_Init

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_Init ( <a href="#"><i>RTC_HandleTypeDef</i></a> * <b>hrtc</b>)</b>                                                                        |
| Function Description | Initializes the RTC peripheral.                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a <i>RTC_HandleTypeDef</i> structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                              |

### 35.2.12 HAL\_RTC\_DeInit

|                      |                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_DeInit ( <i>RTC_HandleTypeDef</i> * hrtc)</b>                                                                                                            |
| Function Description | DeInitializes the RTC peripheral.                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul>                       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul>                       |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• This function doesn't reset the RTC Backup Data registers.</li> <li>• This function does not reset the RTC Backup Data registers.</li> </ul> |

### 35.2.13 HAL\_RTC\_MspInit

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTC_MspInit ( <i>RTC_HandleTypeDef</i> * hrtc)</b>                                                                                                  |
| Function Description | Initializes the RTC MSP.                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 35.2.14 HAL\_RTC\_MspDeInit

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTC_MspDeInit ( <i>RTC_HandleTypeDef</i> * hrtc)</b>                                                                                                |
| Function Description | DeInitializes the RTC MSP.                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

## Notes

- None.

### 35.2.15 HAL\_RTC\_SetTime

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_SetTime (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTime,</code><br><code>uint32_t Format)</code>                                                                                                                                                                                                                                                                                                                                     |
| Function Description | Sets RTC current time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sTime</b> : Pointer to Time structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                |

### 35.2.16 HAL\_RTC\_GetTime

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_GetTime (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTime,</code><br><code>uint32_t Format)</code>                                                                                                                                                                                                                                                                                                                                     |
| Function Description | Gets RTC current time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sTime</b> : Pointer to Time structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• Call HAL_RTC_GetDate() after HAL_RTC_GetTime() to unlock the values in the higher-order calendar shadow registers.</li> </ul>                                                                                                                                                                                                                                                                                                                   |

### 35.2.17 HAL\_RTC\_SetDate

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_SetDate ( <i>RTC_HandleTypeDef</i> * hrtc, <i>RTC_DateTypeDef</i> * sDate, uint32_t Format)</b>                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Sets RTC current date.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sDate</b> : Pointer to date structure</li> <li>• <b>Format</b> : specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                |

### 35.2.18 HAL\_RTC\_GetDate

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_GetDate ( <i>RTC_HandleTypeDef</i> * hrtc, <i>RTC_DateTypeDef</i> * sDate, uint32_t Format)</b>                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Gets RTC current date.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sDate</b> : Pointer to Date structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                |

### 35.2.19 HAL\_RTC\_SetTime

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_SetTime (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTime,</code><br><code>uint32_t Format)</code>                                                                                                                                                                                                                                                                                                                                     |
| Function Description | Sets RTC current time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sTime</b> : Pointer to Time structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                |

### 35.2.20 HAL\_RTC\_SetDate

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_SetDate (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_DateTypeDef * sDate, uint32_t Format)</code>                                                                                                                                                                                                                                                                                                                                                     |
| Function Description | Sets RTC current date.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sDate</b> : Pointer to date structure</li> <li>• <b>Format</b> : specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                |

### 35.2.21 HAL\_RTC\_GetDate

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_GetDate (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_DateTypeDef * sDate,</code><br><code>uint32_t Format)</code>                                                                                                                                                                                                                                                                                                                                     |
| Function Description | Gets RTC current date.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sDate</b> : Pointer to Date structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                |

### 35.2.22 HAL\_RTC\_GetTime

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_GetTime (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTime,</code><br><code>uint32_t Format)</code>                                                                                                                                                                                                                                                                                                                                     |
| Function Description | Gets RTC current time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sTime</b> : Pointer to Time structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• Call HAL_RTC_GetDate() after HAL_RTC_GetTime() to unlock the values in the higher-order calendar shadow registers.</li> </ul>                                                                                                                                                                                                                                                                                                                   |

### 35.2.23 HAL\_RTC\_SetAlarm

|               |                                                                                                                        |
|---------------|------------------------------------------------------------------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_RTC_SetAlarm (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm,</code> |
|---------------|------------------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <b>uint32_t Format)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Function Description | Sets the specified RTC Alarm.                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sAlarm</b> : Pointer to Alarm structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>FORMAT_BIN</b> Binary data format</li> <li>- <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                  |

### 35.2.24 HAL\_RTC\_SetAlarm\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_SetAlarm_IT (</b><br><b>RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm,</b><br><b>uint32_t Format)</b>                                                                                                                                                                                                                                                                                                                                                   |
| Function Description | Sets the specified RTC Alarm with Interrupt.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sAlarm</b> : Pointer to Alarm structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>FORMAT_BIN</b> Binary data format</li> <li>- <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• The Alarm register can only be written when the corresponding Alarm is disabled (Use the <b>HAL_RTC_DeactivateAlarm()</b>).</li> <li>• The <b>HAL_RTC_SetTime()</b> must be called before enabling the Alarm feature.</li> </ul>                                                                                                                                                                                                                  |

### 35.2.25 HAL\_RTC\_DeactivateAlarm

|               |                                                                                                        |
|---------------|--------------------------------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_RTC_DeactivateAlarm (</b><br><b>RTC_HandleTypeDef * hrtc, uint32_t Alarm)</b> |
|---------------|--------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Deactive the specified RTC Alarm.                                                                                                                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>Alarm</b> : Specifies the Alarm. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>RTC_ALARM_A</b> AlarmA</li> <li>– <b>RTC_ALARM_B</b> AlarmB</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                             |

### 35.2.26 HAL\_RTC\_GetAlarm

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_GetAlarm (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm,</code><br><code>uint32_t Alarm, uint32_t Format)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Function Description | Gets the RTC Alarm value and masks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sAlarm</b> : Pointer to Date structure</li> <li>• <b>Alarm</b> : Specifies the Alarm. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>RTC_ALARM_A</b> AlarmA</li> <li>– <b>RTC_ALARM_B</b> AlarmB</li> </ul> </li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

### 35.2.27 HAL\_RTC\_AlarmIRQHandler

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| Function Name        | <code>void HAL_RTC_AlarmIRQHandler ( RTC_HandleTypeDef * hrtc )</code> |
| Function Description | This function handles Alarm interrupt request.                         |

---

|               |                                                                                                                                                                 |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 35.2.28 HAL\_RTC\_PollForAlarmAEvent

|                      |                                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_PollForAlarmAEvent ( RTC_HandleTypeDef * hrtc, uint32_t Timeout)</b>                                                                                                            |
| Function Description | This function handles AlarmA Polling request.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                    |

### 35.2.29 HAL\_RTC\_AlarmAEventCallback

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTC_AlarmAEventCallback ( RTC_HandleTypeDef * hrtc)</b>                                                                                             |
| Function Description | Alarm A callback.                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

Notes

- None.

### 35.2.30 HAL\_RTC\_DeactivateAlarm

|                      |                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_DeactivateAlarm (RTC_HandleTypeDef * hrtc, uint32_t Alarm)</code>                                                                                                                                                                                                                                                                                                 |
| Function Description | Deactive the specified RTC Alarm.                                                                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>Alarm</b> : Specifies the Alarm. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <code>RTC_ALARM_A</code> AlarmA</li> <li>- <code>RTC_ALARM_B</code> AlarmB</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                         |

### 35.2.31 HAL\_RTC\_AlarmIRQHandler

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_RTC_AlarmIRQHandler (RTC_HandleTypeDef * hrtc)</code>                                                                                            |
| Function Description | This function handles Alarm interrupt request.                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 35.2.32 HAL\_RTC\_AlarmAEventCallback

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_RTC_AlarmAEventCallback (RTC_HandleTypeDef * hrtc)</code>                                                                                        |
| Function Description | Alarm A callback.                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 35.2.33 HAL\_RTC\_PollForAlarmAEvent

|                      |                                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_PollForAlarmAEvent (</code><br><code>RTC_HandleTypeDef * hrtc, uint32_t Timeout)</code>                                                                                      |
| Function Description | This function handles AlarmA Polling request.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                    |

### 35.2.34 HAL\_RTC\_SetAlarm

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTC_SetAlarm (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm,</code><br><code>uint32_t Format)</code>                                                                                                                                                                                                                                                                                                                                    |
| Function Description | Sets the specified RTC Alarm.                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sAlarm</b> : Pointer to Alarm structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>FORMAT_BIN</b> Binary data format</li> <li>– <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                  |

### 35.2.35 HAL\_RTC\_SetAlarm\_IT

|               |                                                                                                                           |
|---------------|---------------------------------------------------------------------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_RTC_SetAlarm_IT (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm,</code> |
|---------------|---------------------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <b>uint32_t Format)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Function Description | Sets the specified RTC Alarm with Interrupt.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sAlarm</b> : Pointer to Alarm structure</li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>FORMAT_BIN</b> Binary data format</li> <li>- <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• The Alarm register can only be written when the corresponding Alarm is disabled (Use the HAL_RTC_DeactivateAlarm()).</li> <li>• The HAL_RTC_SetTime() must be called before enabling the Alarm feature.</li> </ul>                                                                                                                                                                                                                                |

### 35.2.36 HAL\_RTC\_GetAlarm

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_GetAlarm (</b><br><b>RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm,</b><br><b>uint32_t Alarm, uint32_t Format)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Function Description | Gets the RTC Alarm value and masks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sAlarm</b> : Pointer to Date structure</li> <li>• <b>Alarm</b> : Specifies the Alarm. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>RTC_ALARM_A</b> AlarmA</li> <li>- <b>RTC_ALARM_B</b> AlarmB</li> </ul> </li> <li>• <b>Format</b> : Specifies the format of the entered parameters. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>FORMAT_BIN</b> Binary data format</li> <li>- <b>FORMAT_BCD</b> BCD data format</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

### 35.2.37 HAL\_RTC\_WaitForSynchro

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_WaitForSynchro (</b><br><i>RTC_HandleTypeDef * hrtc)</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 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"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• The RTC Resynchronization mode is write protected, use the <code>_HAL_RTC_WRITEPROTECTION_DISABLE()</code> before calling this function.</li> <li>• 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.</li> </ul> |

### 35.2.38 HAL\_RTC\_GetState

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_RTCStateTypeDef HAL_RTC_GetState (</b><br><i>RTC_HandleTypeDef * hrtc)</i>                                                                               |
| Function Description | Returns the RTC state.                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 35.2.39 HAL\_RTC\_WaitForSynchro

|                      |                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTC_WaitForSynchro (</b><br><i>RTC_HandleTypeDef * hrtc)</i>                |
| 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"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes         | <ul style="list-style-type: none"> <li>• The RTC Resynchronization mode is write protected, use the <code>_HAL_RTC_WRITEPROTECTION_DISABLE()</code> before calling this function.</li> <li>• 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.</li> </ul> |

## 35.3 RTC Firmware driver defines

### 35.3.1 RTC

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_ALARM_MASK`

*Alarms Definitions*

- `RTC_ALARM_A`
- `RTC_ALARM_B`
- `IS_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\_ASYNCNCH\_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. `__FLAG__`: specifies the RTC Alarm interrupt sources to be enabled or disabled. 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 be enabled or disabled. 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 be enabled or disabled. This parameter can be:  
`RTC_FLAG_ALRAF` `RTC_FLAG_ALRBF`  
**Return value:**None

- **RTC EXTI\_LINE\_ALARM\_EVENT**  
External interrupt line 17 Connected to the RTC Alarm event
  - **RTC EXTI\_LINE\_TAMPER\_TIMESTAMP\_EVENT**  
External interrupt line 19 Connected to the RTC Tamper and Time Stamp events
  - **RTC EXTI\_LINE\_WAKEUPTIMER\_EVENT**  
External interrupt line 20 Connected to the RTC Wakeup event
  - **\_HAL\_RTC\_EXTI\_ENABLE\_IT**  
**Description:** Enable the RTC Exti line.  
**Parameters:** \_\_EXTILINE\_\_: specifies the RTC Exti sources to be enabled or disabled. This parameter can be: RTC\_EXTI\_LINE\_ALARM\_EVENT  
RTC\_EXTI\_LINE\_TAMPER\_TIMESTAMP\_EVENT  
RTC\_EXTI\_LINE\_WAKEUPTIMER\_EVENT  
**Return value:**None:  
    - **\_HAL\_RTC\_ENABLE\_IT**
    - **\_HAL\_RTC\_EXTI\_DISABLE\_IT**
  - **\_HAL\_RTC\_EXTI\_DISABLE\_IT**  
**Description:** Disable the RTC Exti line.  
**Parameters:** \_\_EXTILINE\_\_: specifies the RTC Exti sources to be enabled or disabled. This parameter can be: RTC\_EXTI\_LINE\_ALARM\_EVENT  
RTC\_EXTI\_LINE\_TAMPER\_TIMESTAMP\_EVENT  
RTC\_EXTI\_LINE\_WAKEUPTIMER\_EVENT  
**Return value:**None:  
    - **\_HAL\_RTC\_DISABLE\_IT**
    - **\_HAL\_RTC\_EXTI\_GENERATE\_SWIT**
  - **\_HAL\_RTC\_EXTI\_GENERATE\_SWIT**  
**Description:** Generates a Software interrupt on selected EXTI line.  
**Parameters:** \_\_EXTILINE\_\_: specifies the RTC Exti sources to be enabled or disabled. This parameter can be: RTC\_EXTI\_LINE\_ALARM\_EVENT  
RTC\_EXTI\_LINE\_TAMPER\_TIMESTAMP\_EVENT  
RTC\_EXTI\_LINE\_WAKEUPTIMER\_EVENT  
**Return value:**None:  
    - **\_HAL\_RTC\_EXTI\_CLEAR\_FLAG**
  - **\_HAL\_RTC\_EXTI\_CLEAR\_FLAG**  
**Description:** Clear the RTC Exti flags.  
**Parameters:** \_\_FLAG\_\_: specifies the RTC Exti sources to be enabled or disabled. This parameter can be: RTC\_EXTI\_LINE\_ALARM\_EVENT  
RTC\_EXTI\_LINE\_TAMPER\_TIMESTAMP\_EVENT  
RTC\_EXTI\_LINE\_WAKEUPTIMER\_EVENT  
**Return value:**None:  
    - **\_HAL\_RTC\_CLEAR\_FLAG**
- 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*

- **FORMAT\_BIN**
- **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\_SEPTEMBER**
- **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\_PUSHPULL**
- **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**

## 36 HAL RTC Extension Driver

### 36.1 RTCEx Firmware driver registers structures

#### 36.1.1 RTC\_TamperTypeDef

*RTC\_TamperTypeDef* is defined in the `stm32l1xx_hal_rtc_ex.h`

##### 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\*](#)

#### 36.1.2 RTC\_TimeTypeDef

*RTC\_TimeTypeDef* is defined in the `stm32l1xx_hal_rtc_ex.h`

##### Data Fields

- *uint8\_t Hours*
- *uint8\_t Minutes*
- *uint8\_t Seconds*
- *uint32\_t SubSeconds*
- *uint8\_t TimeFormat*
- *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.
- ***uint32\_t RTC\_TimeTypeDef::SubSeconds*** Specifies the RTC Time SubSeconds. 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::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 BCK bit in CR register to store the operation. This parameter can be a value of [RTC\\_StoreOperation\\_Definitions](#).

### 36.1.3 RTC\_AlarmTypeDef

*RTC\_AlarmTypeDef* is defined in the `stm32l1xx_hal_rtc_ex.h`

#### 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](#).

## 36.2 RTCEEx Firmware driver API description

The following section lists the various functions of the RTCEEx library.

### 36.2.1 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.

### 36.2.2 RTC TimeStamp and Tamper functions

This section provides functions allowing to configure TimeStamp feature

- [\*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()`

### 36.2.3 RTC Wake-up functions

This section provides functions allowing to configure Wake-up feature

- `HAL_RTCEEx_SetWakeUpTimer()`
- `HAL_RTCEEx_SetWakeUpTimer_IT()`
- `HAL_RTCEEx_DeactivateWakeUpTimer()`
- `HAL_RTCEEx_GetWakeUpTimer()`
- `HAL_RTCEEx_WakeUpTimerIRQHandler()`
- `HAL_RTCEEx_WakeUpTimerEventCallback()`
- `HAL_RTCEEx_PollForWakeUpTimerEvent()`

### 36.2.4 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.
- `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_RTCEx_SetRefClock()`
- `HAL_RTCEx_DeactivateRefClock()`
- `HAL_RTCEx_EnableBypassShadow()`
- `HAL_RTCEx_DisableBypassShadow()`

### 36.2.5 Extended features functions

This section provides functions allowing to:

- RTC Alram B callback
- RTC Poll for Alarm B request
- `HAL_RTCEx_AlarmBEventCallback()`
- `HAL_RTCEx_PollForAlarmBEvent()`

### 36.2.6 HAL\_RTCEx\_SetTimeStamp

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_SetTimeStamp (RTC_HandleTypeDef * hrtc, uint32_t TimeStampEdge)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Function Description | Sets TimeStamp.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>TimeStampEdge</b> : Specifies the pin edge on which the TimeStamp is activated. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>RTC_TIMESTAMPEDGE_RISING</b> the Time stamp event occurs on the rising edge of the related pin.</li> <li>- <b>RTC_TIMESTAMPEDGE_FALLING</b> the Time stamp event occurs on the falling edge of the related pin.</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• This API must be called before enabling the TimeStamp feature.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

### 36.2.7 HAL\_RTCEx\_SetTimeStamp\_IT

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_SetTimeStamp_IT (RTC_HandleTypeDef * hrtc, uint32_t TimeStampEdge)</code>                                                     |
| Function Description | Sets TimeStamp with Interrupt.                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>TimeStampEdge</b> : Specifies the pin edge on which the</li> </ul>                                                  |

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.

### 36.2.8 HAL\_RTCEx\_DeactivateTimeStamp

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_DeactivateTimeStamp (</code><br><code>RTC_HandleTypeDef * hrtc)</code>                                                        |
| Function Description | Deactivates TimeStamp.                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.9 HAL\_RTCEx\_GetTimeStamp

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_GetTimeStamp (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef *</code><br><code>sTimeStamp, RTC_DateTypeDef * sTimeStampDate, uint32_t</code><br><code>Format)</code>                                                                                                                                                                                                                                                             |
| Function Description | Gets the RTC TimeStamp value.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>sTimeStamp</b> : Pointer to Time structure</li> <li>• <b>sTimeStampDate</b> : Pointer to Date structure</li> <li>• <b>Format</b> : specifies the format of the entered parameters. This parameter can be one of the following values: FORMAT_BIN: Binary data format FORMAT_BCD: BCD data format</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                          |

## Notes

- None.

### 36.2.10 HAL\_RTCEx\_SetTamper

|                      |                                                                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_SetTamper (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_TamperTypeDef * sTamper)</code>                                                                                           |
| Function Description | Sets Tamper.                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li><li>• <b>sTamper</b> : Pointer to Tamper Structure.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• By calling this API we disable the tamper interrupt for all tampers.</li></ul>                                                                                                |

### 36.2.11 HAL\_RTCEx\_SetTamper\_IT

|                      |                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_SetTamper_IT (</code><br><code>RTC_HandleTypeDef * hrtc, RTC_TamperTypeDef * sTamper)</code>                                                                                  |
| Function Description | Sets Tamper with interrupt.                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li><li>• <b>sTamper</b> : Pointer to RTC Tamper.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"><li>• By calling this API we force the tamper interrupt for all tampers.</li></ul>                                                                                            |

### 36.2.12 HAL\_RTCEx\_DeactivateTamper

|               |                                                             |
|---------------|-------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_RTCEx_DeactivateTamper (</code> |
|---------------|-------------------------------------------------------------|

---

***RTC\_HandleTypeDef \* hrtc, uint32\_t Tamper)***

|                      |                                                                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Deactivates Tamper.                                                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>Tamper</b> : Selected tamper pin. This parameter can be a value of Tamper Pins Definitions</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                |

### 36.2.13 HAL\_RTCEx\_TamperTimeStampIRQHandler

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTCEx_TamperTimeStampIRQHandler (</b><br><b><i>RTC_HandleTypeDef * hrtc)</i></b>                                                                    |
| Function Description | This function handles TimeStamp interrupt request.                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.14 HAL\_RTCEx\_TimeStampEventCallback

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTCEx_TimeStampEventCallback (</b><br><b><i>RTC_HandleTypeDef * hrtc)</i></b>                                                                       |
| Function Description | TimeStamp callback.                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.15 HAL\_RTCEx\_Tamper1EventCallback

|                      |                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTCEx_Tamper1EventCallback (</b><br><i>RTC_HandleTypeDef</i> * hrtc)                                                                              |
| Function Description | Tamper 1 callback.                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                       |

### 36.2.16 HAL\_RTCEx\_Tamper2EventCallback

|                      |                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTCEx_Tamper2EventCallback (</b><br><i>RTC_HandleTypeDef</i> * hrtc)                                                                              |
| Function Description | Tamper 2 callback.                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                       |

### 36.2.17 HAL\_RTCEx\_Tamper3EventCallback

|                      |                                                                                  |
|----------------------|----------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTCEx_Tamper3EventCallback (</b><br><i>RTC_HandleTypeDef</i> * hrtc) |
| Function Description | Tamper 3 callback.                                                               |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : RTC handle</li></ul>       |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                          |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                          |

### 36.2.18 HAL\_RTCEx\_PollForTimeStampEvent

|                      |                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_PollForTimeStampEvent (</code><br><code>RTC_HandleTypeDef * hrtc, uint32_t Timeout)</code>                                                                              |
| Function Description | This function handles TimeStamp polling request.                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                   |

### 36.2.19 HAL\_RTCEx\_PollForTamper1Event

|                      |                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_PollForTamper1Event (</code><br><code>RTC_HandleTypeDef * hrtc, uint32_t Timeout)</code>                                                                                |
| Function Description | This function handles Tamper1 Polling.                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                   |

### 36.2.20 HAL\_RTCEx\_PollForTamper2Event

|                      |                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_PollForTamper2Event (</code><br><code>RTC_HandleTypeDef * hrtc, uint32_t Timeout)</code>                                                                                |
| Function Description | This function handles Tamper2 Polling.                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |

- |               |                                                                     |
|---------------|---------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul> |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>             |

### 36.2.21 HAL\_RTCEx\_PollForTamper3Event

|                      |                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_PollForTamper3Event (</b><br><i>RTC_HandleTypeDef</i> * hrtc, uint32_t Timeout)         |
| Function Description | This function handles Tamper3 Polling.                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : RTC handle</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                |

### 36.2.22 HAL\_RTCEx\_SetWakeUpTimer

|                      |                                                                                                                                                                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_SetWakeUpTimer (</b><br><i>RTC_HandleTypeDef</i> * hrtc, uint32_t WakeUpCounter,<br>uint32_t WakeUpClock)                                                                                                                    |
| Function Description | Sets wake up timer.                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li><li>• <b>WakeUpCounter</b> : Wake up counter</li><li>• <b>WakeUpClock</b> : Wake up clock</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                     |

### 36.2.23 HAL\_RTCEx\_SetWakeUpTimer\_IT

|                      |                                                                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_SetWakeUpTimer_IT (</code><br><code>RTC_HandleTypeDef * hrtc, uint32_t WakeUpCounter,</code><br><code>uint32_t WakeUpClock)</code>                                                                                            |
| Function Description | Sets wake up timer with interrupt.                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>WakeUpCounter</b> : Wake up counter</li> <li>• <b>WakeUpClock</b> : Wake up clock</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                       |

### 36.2.24 HAL\_RTCEx\_DeactivateWakeUpTimer

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_RTCEx_DeactivateWakeUpTimer (</code><br><code>RTC_HandleTypeDef * hrtc)</code>                                                               |
| Function Description | Deactivates wake up timer counter.                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                           |

### 36.2.25 HAL\_RTCEx\_GetWakeUpTimer

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_RTCEx_GetWakeUpTimer (</code><br><code>RTC_HandleTypeDef * hrtc)</code>                                                                      |
| Function Description | Gets wake up timer counter.                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Counter value</b></li> </ul>                                                                                        |

### 36.2.26 HAL\_RTCEx\_WakeUpTimerIRQHandler

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_RTCEx_WakeUpTimerIRQHandler (RTC_HandleTypeDef * hrtc)</code>                                                                                    |
| Function Description | This function handles Wake Up Timer interrupt request.                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.27 HAL\_RTCEx\_WakeUpTimerEventCallback

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_RTCEx_WakeUpTimerEventCallback (RTC_HandleTypeDef * hrtc)</code>                                                                                 |
| Function Description | Wake Up Timer callback.                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.28 HAL\_RTCEx\_PollForWakeUpTimerEvent

|                      |                                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_RTCEx_PollForWakeUpTimerEvent (RTC_HandleTypeDef * hrtc, uint32_t Timeout)</code>                                                                                                |
| Function Description | This function handles Wake Up Timer Polling.                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                    |

### 36.2.29 HAL\_RTCEx\_BKUPWrite

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_RTCEx_BKUPWrite ( RTC_HandleTypeDef * hrtc,<br/>uint32_t BackupRegister, uint32_t Data)</code>                                                                                                                                                                                                                                                                                                      |
| Function Description | Writes a data in a specified RTC Backup data register.                                                                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>BackupRegister</b> : RTC Backup data Register number. This parameter can be: RTC_BKP_DRx where x can be from 0 to 19 to specify the register.</li> <li>• <b>Data</b> : Data to be written in the specified RTC Backup data register.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                          |

### 36.2.30 HAL\_RTCEx\_BKUPRead

|                      |                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_RTCEx_BKUPRead ( RTC_HandleTypeDef * hrtc, uint32_t BackupRegister)</code>                                                                                                                                                                                                                               |
| Function Description | Reads data from the specified RTC Backup data Register.                                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>BackupRegister</b> : RTC Backup data Register number. This parameter can be: RTC_BKP_DRx where x can be from 0 to 19 to specify the register.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Read value</b></li> </ul>                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                   |

### 36.2.31 HAL\_RTCEx\_SetCoarseCalib

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_SetCoarseCalib (</b><br><b>RTC_HandleTypeDef * hrtc, uint32_t CalibSign, uint32_t</b><br><b>Value)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Sets the Coarse calibration parameters.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>CalibSign</b> : Specifies the sign of the coarse calibration value. This parameter can be one of the following values : <ul style="list-style-type: none"> <li>- <b>RTC_CALIBSIGN_POSITIVE</b> The value sign is positive</li> <li>- <b>RTC_CALIBSIGN_NEGATIVE</b> The value sign is negative</li> </ul> </li> <li>• <b>Value</b> : value of coarse calibration expressed in ppm (coded on 5 bits).</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• This Calibration value should be between 0 and 63 when using negative sign with a 2-ppm step.</li> <li>• This Calibration value should be between 0 and 126 when using positive sign with a 4-ppm step.</li> </ul>                                                                                                                                                                                                                                                                                                                       |

### 36.2.32 HAL\_RTCEx\_DeactivateCoarseCalib

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_DeactivateCoarseCalib (</b><br><b>RTC_HandleTypeDef * hrtc)</b>                                                                  |
| Function Description | Deactivates the Coarse calibration parameters.                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                             |
| Notes                | • None.                                                                                                                                                         |

### 36.2.33 HAL\_RTCEx\_SetSmoothCalib

|               |                                                                                                                                                                                                               |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_RTCEx_SetSmoothCalib (</b><br><b>RTC_HandleTypeDef * hrtc, uint32_t SmoothCalibPeriod,</b><br><b>uint32_t SmoothCalibPlusPulses, uint32_t</b><br><b>SmoothCalibMinusPulsesValue)</b> |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Sets the Smooth calibration parameters.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>SmoothCalibPeriod</b> : Select the Smooth Calibration Period. This parameter can be one of the following values : <ul style="list-style-type: none"> <li>– <b>RTC_SMOOTHCALIB_PERIOD_32SEC</b> The smooth calibration period is 32s.</li> <li>– <b>RTC_SMOOTHCALIB_PERIOD_16SEC</b> The smooth calibration period is 16s.</li> <li>– <b>RTC_SMOOTHCALIB_PERIOD_8SEC</b> The smooth calibration period is 8s.</li> </ul> </li> <li>• <b>SmoothCalibPlusPulses</b> : Select to Set or reset the CALP bit. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>RTC_SMOOTHCALIB_PLUSPULSES_SET</b> Add one RTCCLK pulse every 2*11 pulses.</li> <li>– <b>RTC_SMOOTHCALIB_PLUSPULSES_RESET</b> No RTCCLK pulses are added.</li> </ul> </li> <li>• <b>SmoothCalibMinusPulsesValue</b> : Select the value of CALM[8:0] bits. This parameter can be one any value from 0 to 0x000001FF.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• To deactivate the smooth calibration, the field SmoothCalibPlusPulses must be equal to SMOOTHCALIB_PLUSPULSES_RESET and the field SmoothCalibMinusPulsesValue must be equal to 0.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

### 36.2.34 HAL\_RTCEx\_SetSynchroShift

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_SetSynchroShift (</b><br><b>RTC_HandleTypeDef * hrtc, uint32_t ShiftAdd1S, uint32_t</b><br><b>ShiftSubFS)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Function Description | Configures the Synchronization Shift Control Settings.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>ShiftAdd1S</b> : 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"> <li>– <b>RTC_SHIFTADD1S_SET</b> Add one second to the clock calendar.</li> <li>– <b>RTC_SHIFTADD1S_RESET</b> No effect.</li> </ul> </li> <li>• <b>ShiftSubFS</b> : Select the number of Second Fractions to substitute. This parameter can be one any value from 0 to 0x7FFF.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

- |       |                                                                                                                             |
|-------|-----------------------------------------------------------------------------------------------------------------------------|
| Notes | <ul style="list-style-type: none"> <li>• When REFCKON is set, firmware must not write to Shift control register.</li> </ul> |
|-------|-----------------------------------------------------------------------------------------------------------------------------|

### 36.2.35 HAL\_RTCEx\_SetCalibrationOutPut

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_SetCalibrationOutPut (</b><br><b>RTC_HandleTypeDef * hrtc, uint32_t CalibOutput)</b>                                                                                                                                                                                                                                                                                                                                                                                      |
| Function Description | Configures the Calibration Pinout (RTC_CALIB) Selection (1Hz or 512Hz).                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>CalibOutput</b> : : Select the Calibration output Selection . This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>RTC_CALIBOUTPUT_512HZ</b> A signal has a regular waveform at 512Hz.</li> <li>– <b>RTC_CALIBOUTPUT_1HZ</b> A signal has a regular waveform at 1Hz.</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                |

### 36.2.36 HAL\_RTCEx\_DeactivateCalibrationOutPut

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef</b><br><b>HAL_RTCEx_DeactivateCalibrationOutPut (</b><br><b>RTC_HandleTypeDef * hrtc)</b>                                                  |
| Function Description | Deactivates the Calibration Pinout (RTC_CALIB) Selection (1Hz or 512Hz).                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.37 HAL\_RTCEx\_SetRefClock

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_SetRefClock (</b><br><b><i>RTC_HandleTypeDef</i> * hrtc)</b>                                                                     |
| Function Description | Enables the RTC reference clock detection.                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.38 HAL\_RTCEx\_DeactivateRefClock

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_DeactivateRefClock (</b><br><b><i>RTC_HandleTypeDef</i> * hrtc)</b>                                                              |
| Function Description | Disable the RTC reference clock detection.                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.39 HAL\_RTCEx\_EnableBypassShadow

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_EnableBypassShadow (</b><br><b><i>RTC_HandleTypeDef</i> * hrtc)</b>                                                              |
| Function Description | Enables the Bypass Shadow feature.                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• When the Bypass Shadow is enabled the calendar value are taken directly from the Calendar counter.</li> </ul>          |

### 36.2.40 HAL\_RTCEx\_DisableBypassShadow

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_DisableBypassShadow (</b><br><b>RTC_HandleTypeDef * hrtc)</b>                                                                    |
| Function Description | Disables the Bypass Shadow feature.                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• When the Bypass Shadow is enabled the calendar value are taken directly from the Calendar counter.</li> </ul>          |

### 36.2.41 HAL\_RTCEx\_AlarmBEventCallback

|                      |                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_RTCEx_AlarmBEventCallback (</b><br><b>RTC_HandleTypeDef * hrtc)</b>                                                                                 |
| Function Description | Alarm B callback.                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                       |

### 36.2.42 HAL\_RTCEx\_PollForAlarmBEvent

|                      |                                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_RTCEx_PollForAlarmBEvent (</b><br><b>RTC_HandleTypeDef * hrtc, uint32_t Timeout)</b>                                                                                                |
| Function Description | This function handles AlarmB Polling request.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hrtc</b> : pointer to a RTC_HandleTypeDef structure that contains the configuration information for RTC.</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |

---

|               |                     |
|---------------|---------------------|
| Return values | • <b>HAL status</b> |
| Notes         | • None.             |

## 36.3 RTCEEx Firmware driver defines

### 36.3.1 RTCEEx

RTCEEx

*Add 1 Second Parameter Definitions*

- RTC\_SHIFTADD1S\_RESET
- RTC\_SHIFTADD1S\_SET
- IS\_RTC\_SHIFT\_ADD1S

*Backup Registers Definitions*

- 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

*RTCEEx Exported Macros*

- \_\_HAL\_RTC\_WAKEUPTIMER\_ENABLE  
**Description:** Enable the RTC WakeUp Timer peripheral.  
**Parameters:** \_\_HANDLE\_\_: specifies the RTC handle.  
**Return value:**None
- \_\_HAL\_RTC\_TIMESTAMP\_ENABLE  
**Description:** Enable the RTC TimeStamp peripheral.  
**Parameters:** \_\_HANDLE\_\_: specifies the RTC handle.  
**Return value:**None
- \_\_HAL\_RTC\_WAKEUPTIMER\_DISABLE  
**Description:** Disable the RTC WakeUp Timer peripheral.  
**Parameters:** \_\_HANDLE\_\_: specifies the RTC handle.  
**Return value:**None
- \_\_HAL\_RTC\_TIMESTAMP\_DISABLE  
**Description:** Disable the RTC TimeStamp peripheral.  
**Parameters:** \_\_HANDLE\_\_: specifies the RTC handle.  
**Return value:**None
- \_\_HAL\_RTC\_COARSE\_CALIB\_ENABLE  
**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_TAMPER_GET_IT`**  
**Description:** Check whether the specified RTC Tamper interrupt has occurred or not.  
**Parameters:** `_HANDLE_`: specifies the RTC handle. `_FLAG_`: specifies the RTC Tamper interrupt sources to be enabled or disabled. This parameter can be:  
RTC\_IT\_TAMP1 RTC\_IT\_TAMP2 RTC\_IT\_TAMP3  
**Return value:**None
- **`_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\_TIMESTAMP\_GET\_IT**

**Description:** Check whether the specified RTC TimeStamp interrupt has occurred or not.

**Parameters:** `_HANDLE_`: specifies the RTC handle. `_FLAG_`: 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\_TIMESTAMP\_GET\_FLAG**

**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`

**Return value:**None

- **\_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`

**Return value:**None

- **\_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 Tamper Flag sources to be enabled or disabled. This parameter can be:

`RTC_FLAG_WUTF`

**Return value:**None

#### ***Output selection Definitions***

- `RTC_OUTPUT_DISABLE`
- `RTC_OUTPUT_ALARMA`
- `RTC_OUTPUT_ALARMNB`

- **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\_TAMPER\_FILTER**

***Tamper Pins Definitions***

- **RTC\_TAMPER\_1**
- **RTC\_TAMPER\_2**
- **RTC\_TAMPER\_3**
- **IS\_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\_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\_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\_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\_TAMPER\_TIMESTAMPONTAMPER\_DETECTION**

#### *Tamper Trigger Definitions*

- **RTC\_TAMPERTRIGGER\_RISINGEDGE**
- **RTC\_TAMPERTRIGGER\_FALLINGEDGE**
- **RTC\_TAMPERTRIGGER\_LOWLEVEL**
- **RTC\_TAMPERTRIGGER\_HIGHLEVEL**
- **IS\_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\_WAKEUP\_CLOCK**
- **IS\_WAKEUP\_COUNTER**

## 37 HAL SD Generic Driver

### 37.1 SD Firmware driver registers structures

#### 37.1.1 SD\_HandleTypeDef

*SD\_HandleTypeDef* is defined in the `stm32l1xx_hal_sd.h`

##### 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

#### 37.1.2 HAL\_SD\_CSDTypeDef

*HAL\_SD\_CSDTypeDef* is defined in the `stm32l1xx_hal_sd.h`

##### 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::ContentProtectAppli` 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

### 37.1.3 HAL\_SD\_CIDTypedef

`HAL_SD_CIDTypedef` is defined in the `stm32l1xx_hal_sd.h`

#### Data Fields

- `_IO uint8_t ManufacturerID`
- `_IO uint16_t OEM_Applid`
- `_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_Applid` 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

### 37.1.4 HAL\_SD\_CardStatusTypedef

`HAL_SD_CardStatusTypedef` is defined in the `stm32l1xx_hal_sd.h`

#### 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

### 37.1.5 HAL\_SD\_CardInfoTypedef

`HAL_SD_CardInfoTypedef` is defined in the `stm32l1xx_hal_sd.h`

**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

## 37.2 SD Firmware driver API description

The following section lists the various functions of the SD library.

### 37.2.1 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 `_SDIO_CLK_ENABLE()`;
  - b. SDIO pins configuration for SD card
    - Enable the clock for the SDIO GPIOs using the functions `_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 `_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-byte 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.
- You can read from SD card in DMA mode by using function `HAL_SD_ReadBlocks_DMA()`. This function support only 512-byte 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_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-byte 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.

- You can write to SD card in DMA mode by using function `HAL_SD_WriteBlocks_DMA()`. This function support only 512-byte 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

## 37.2.2 Initialization and de-initialization functions

This section provides functions allowing to initialize/de-initialize the SD card device to be ready for use.

- `HAL_SD_Init()`
- `HAL_SD_DelInit()`
- `HAL_SD_MspInit()`
- `HAL_SD_MspDelInit()`

## 37.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the data transfer from/to SD card.

- `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()`

### 37.2.4 Peripheral Control functions

This subsection provides a set of functions allowing to control the SD card operations.

- [\*HAL\\_SD\\_Get\\_CardInfo\(\)\*](#)
- [\*HAL\\_SD\\_WideBusOperation\\_Config\(\)\*](#)
- [\*HAL\\_SD\\_StopTransfer\(\)\*](#)
- [\*HAL\\_SD\\_HighSpeed\(\)\*](#)

### 37.2.5 Peripheral State functions

This subsection permits to get in runtime the status of the peripheral and the data flow.

- [\*HAL\\_SD\\_SendSDStatus\(\)\*](#)
- [\*HAL\\_SD\\_GetStatus\(\)\*](#)
- [\*HAL\\_SD\\_GetCardStatus\(\)\*](#)

### 37.2.6 HAL\_SD\_Init

|                      |                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef</b> <b>HAL_SD_Init</b> ( <b>SD_HandleTypeDef</b> * <b>hsd</b> , <b>HAL_SD_CardInfoTypedef</b> * <b>SDCardInfo</b> )                                |
| Function Description | Initializes the SD card according to the specified parameters in the <b>SD_HandleTypeDef</b> and create the associated handle.                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>SDCardInfo</b> : <b>HAL_SD_CardInfoTypedef</b> structure for SD card information</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL SD error state</b></li> </ul>                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                 |

### 37.2.7 HAL\_SD\_DelInit

|                      |                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef</b> <b>HAL_SD_DelInit</b> ( <b>SD_HandleTypeDef</b> * <b>hsd</b> ) |
| Function Description | De-Initializes the SD card.                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> </ul>              |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                   |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                               |

### 37.2.8 HAL\_SD\_MspInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_MspInit ( <i>SD_HandleTypeDef</i> * hsd)</b>                |
| Function Description | Initializes the SD MSP.                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                  |

### 37.2.9 HAL\_SD\_MspDeInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_MspDeInit ( <i>SD_HandleTypeDef</i> * hsd)</b>              |
| Function Description | De-Initialize SD MSP.                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                  |

### 37.2.10 HAL\_SD\_ReadBlocks

|                      |                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_ReadBlocks ( <i>SD_HandleTypeDef</i> * hsd, uint32_t * pReadBuffer, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumberOfBlocks)</b>                                                                                                                                                                                                                     |
| Function Description | Reads block(s) from a specified address in a card.                                                                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>pReadBuffer</b> : pointer to the buffer that will contain the received data</li> <li>• <b>ReadAddr</b> : Address from where data is to be read</li> <li>• <b>BlockSize</b> : SD card Data block size This parameter should be 512</li> <li>• <b>NumberOfBlocks</b> : Number of SD blocks to read</li> </ul> |

- |               |                                                                                |
|---------------|--------------------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"> <li>• <b>SD Card error state</b></li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                      |

### 37.2.11 HAL\_SD\_WriteBlocks

|                      |                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_WriteBlocks (</b><br><b><i>SD_HandleTypeDef</i> * hsd, uint32_t * pWriteBuffer, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumberOfBlocks)</b>                                                                                                                                                                                                                  |
| Function Description | Allows to write block(s) to a specified address in a card.                                                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>pWriteBuffer</b> : pointer to the buffer that will contain the data to transmit</li> <li>• <b>WriteAddr</b> : Address from where data is to be written</li> <li>• <b>BlockSize</b> : SD card Data block size This parameter should be 512.</li> <li>• <b>NumberOfBlocks</b> : Number of SD blocks to write</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>SD Card error state</b></li> </ul>                                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                           |

### 37.2.12 HAL\_SD\_ReadBlocks\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_ReadBlocks_DMA (</b><br><b><i>SD_HandleTypeDef</i> * hsd, uint32_t * pReadBuffer, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumberOfBlocks)</b>                                                                                                                                                                                                      |
| Function Description | Reads block(s) from a specified address in a card.                                                                                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>pReadBuffer</b> : Pointer to the buffer that will contain the received data</li> <li>• <b>ReadAddr</b> : Address from where data is to be read</li> <li>• <b>BlockSize</b> : SD card Data block size This parameter should be 512.</li> <li>• <b>NumberOfBlocks</b> : Number of blocks to read.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>SD Card error state</b></li> </ul>                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• This API should be followed by the function <b>HAL_SD_CheckReadOperation()</b> to check the completion of</li> </ul>                                                                                                                                                                                                                            |

---

the read process

### 37.2.13 HAL\_SD\_WriteBlocks\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_SD_ErrorTypeDef HAL_SD_WriteBlocks_DMA (</code><br><code>SD_HandleTypeDef * hsd, uint32_t * pWriteBuffer, uint64_t</code><br><code>WriteAddr, uint32_t BlockSize, uint32_t NumberOfBlocks)</code>                                                                                                                                                                                       |
| Function Description | Writes block(s) to a specified address in a card.                                                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>pWriteBuffer</b> : pointer to the buffer that will contain the data to transmit</li> <li>• <b>WriteAddr</b> : Address from where data is to be read</li> <li>• <b>BlockSize</b> : the SD card Data block size This parameter should be 512.</li> <li>• <b>NumberOfBlocks</b> : Number of blocks to write</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>SD Card error state</b></li> </ul>                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• This API should be followed by the function <code>HAL_SD_CheckWriteOperation()</code> to check the completion of the write process (by SD current status polling).</li> </ul>                                                                                                                                                                            |

### 37.2.14 HAL\_SD\_CheckReadOperation

|                      |                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_SD_ErrorTypeDef HAL_SD_CheckReadOperation (</code><br><code>SD_HandleTypeDef * hsd, uint32_t Timeout)</code>  |
| Function Description | This function waits until the SD DMA data read transfer is finished.                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>SD Card error state</b></li> </ul>                                          |

Notes

- None.

### 37.2.15 HAL\_SD\_CheckWriteOperation

|                      |                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_CheckWriteOperation ( <i>SD_HandleTypeDef</i> * hsd, uint32_t Timeout)</b>                |
| Function Description | This function waits until the SD DMA data write transfer is finished.                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <b>SD Card error state</b>                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                               |

### 37.2.16 HAL\_SD\_Erase

|                      |                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_Erase ( <i>SD_HandleTypeDef</i> * hsd, uint64_t startaddr, uint64_t endaddr)</b>                                                           |
| Function Description | Erases the specified memory area of the given SD card.                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>startaddr</b> : Start byte address</li> <li>• <b>endaddr</b> : End byte address</li> </ul> |
| Return values        | <b>SD Card error state</b>                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                |

### 37.2.17 HAL\_SD\_IRQHandler

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_IRQHandler ( <i>SD_HandleTypeDef</i> * hsd)</b>             |
| Function Description | This function handles SD card interrupt request.                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                  |

### 37.2.18 HAL\_SD\_XferCpltCallback

|                      |                                                                          |
|----------------------|--------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_XferCpltCallback ( <i>SD_HandleTypeDef</i> * hsd)</b>     |
| Function Description | SD end of transfer callback.                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsd</b> : SD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                  |

### 37.2.19 HAL\_SD\_XferErrorCallback

|                      |                                                                          |
|----------------------|--------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_XferErrorCallback ( <i>SD_HandleTypeDef</i> * hsd)</b>    |
| Function Description | SD Transfer Error callback.                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsd</b> : SD handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                  |

### 37.2.20 HAL\_SD\_DMA\_RxCpltCallback

|                      |                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_DMA_RxCpltCallback ( <i>DMA_HandleTypeDef</i> * hdma)</b>                                                                                                                  |
| Function Description | SD Transfer complete Rx callback in non blocking mode.                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdma</b> : pointer to a <i>DMA_HandleTypeDef</i> structure that contains the configuration information for the specified DMA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                   |

### 37.2.21 HAL\_SD\_DMA\_RxErrorCallback

|                      |                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_DMA_RxErrorCallback ( DMA_HandleTypeDef * hdma)</b>                                                                                                                 |
| Function Description | SD DMA transfer complete Rx error callback.                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdma</b> : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                            |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                            |

### 37.2.22 HAL\_SD\_DMA\_TxCpltCallback

|                      |                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_DMA_TxCpltCallback ( DMA_HandleTypeDef * hdma)</b>                                                                                                                  |
| Function Description | SD Transfer complete Tx callback in non blocking mode.                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdma</b> : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                            |

### 37.2.23 HAL\_SD\_DMA\_TxErrorCallback

|                      |                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SD_DMA_TxErrorCallback ( DMA_HandleTypeDef * hdma)</b>                                                                                                         |
| Function Description | SD DMA transfer complete error Tx callback.                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdma</b> : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA</li></ul> |

---

|               |                                                         |
|---------------|---------------------------------------------------------|
|               | module.                                                 |
| Return values | <ul style="list-style-type: none"> <li>None.</li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>None.</li> </ul> |

### 37.2.24 HAL\_SD\_Get\_CardInfo

|                      |                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_Get_CardInfo (</b><br><b><i>SD_HandleTypeDef</i> * hsd, <i>HAL_SD_CardInfoTypedef</i> *</b><br><b><i>pCardInfo</i>)</b>                                          |
| Function Description | Returns information about specific card.                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li><b>hsd</b> : SD handle</li> <li><b>pCardInfo</b> : Pointer to a <i>HAL_SD_CardInfoTypedef</i> structure that contains all SD cardinformation</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>SD Card error state</b></li> </ul>                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                        |

### 37.2.25 HAL\_SD\_WideBusOperation\_Config

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_WideBusOperation_Config (</b><br><b><i>SD_HandleTypeDef</i> * hsd, <i>uint32_t</i> WideMode)</b>                                                                                                                                                                                                                                                                                           |
| Function Description | Enables wide bus operation for the requested card if supported by card.                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li><b>hsd</b> : SD handle</li> <li><b>WideMode</b> : Specifies the SD card wide bus mode This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>SDIO_BUS_WIDE_8B</b> 8-bit data transfer (Only for MMC)</li> <li>– <b>SDIO_BUS_WIDE_4B</b> 4-bit data transfer</li> <li>– <b>SDIO_BUS_WIDE_1B</b> 1-bit data transfer</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>SD Card error state</b></li> </ul>                                                                                                                                                                                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                  |

### 37.2.26 HAL\_SD\_StopTransfer

|                      |                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_StopTransfer (</b><br><b><i>SD_HandleTypeDef * hsd</i></b> ) |
| Function Description | Aborts an ongoing data transfer.                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsd</b> : SD handle</li></ul>                   |
| Return values        | <ul style="list-style-type: none"><li>• <b>SD Card error state</b></li></ul>               |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                    |

### 37.2.27 HAL\_SD\_HighSpeed

|                      |                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_HighSpeed (</b><br><b><i>SD_HandleTypeDef * hsd</i></b> )                                                                    |
| Function Description | Switches the SD card to High Speed mode.                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsd</b> : SD handle</li></ul>                                                                                   |
| Return values        | <ul style="list-style-type: none"><li>• <b>SD Card error state</b></li></ul>                                                                               |
| Notes                | <ul style="list-style-type: none"><li>• This operation should be followed by the configuration of PLL to have SDIOCK clock between 67 and 75 MHz</li></ul> |

### 37.2.28 HAL\_SD\_SendSDStatus

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypeDef HAL_SD_SendSDStatus (</b><br><b><i>SD_HandleTypeDef * hsd, uint32_t * pSDstatus</i></b> )                                                                     |
| Function Description | Returns the current SD card's status.                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsd</b> : SD handle</li><li>• <b>pSDstatus</b> : Pointer to the buffer that will contain the SD card status SD Status register)</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>SD Card error state</b></li></ul>                                                                                                         |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                              |

### 37.2.29 HAL\_SD\_GetStatus

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_TransferStateTypedef HAL_SD_GetStatus (</b><br><b><i>SD_HandleTypeDef</i> * hsd)</b> |
| Function Description | Gets the current sd card data status.                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> </ul>                     |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Data Transfer state</b></li> </ul>                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                      |

### 37.2.30 HAL\_SD\_GetCardStatus

|                      |                                                                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SD_ErrorTypedef HAL_SD_GetCardStatus (</b><br><b><i>SD_HandleTypeDef</i> * hsd, <i>HAL_SD_CardStatusTypedef</i> * pCardStatus)</b>                                                                            |
| Function Description | Gets the SD card status.                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsd</b> : SD handle</li> <li>• <b>pCardStatus</b> : Pointer to the <i>HAL_SD_CardStatusTypedef</i> structure that will contain the SD card status information</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>SD Card error state</b></li> </ul>                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                            |

## 37.3 SD Firmware driver defines

### 37.3.1 SD

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 line.
- **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**
- **SD\_CMD\_GO\_INACTIVE\_STATE**  
Sends an addressed card into the inactive state.
- **SD\_CMD\_SET\_BLOCKLEN**  
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\_READ\_SINGLE\_BLOCK**  
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\_MULT\_BLOCK**  
Continuously transfers data blocks from card to host until interrupted by STOP\_TRANSMISSION command.
- **SD\_CMD\_HS\_BUSTEST\_WRITE**  
64 bytes tuning pattern is sent for SDR50 and SDR104.
- **SD\_CMD\_WRITE\_DAT\_UNTIL\_STOP**  
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**  
**Return value:**None
- **\_\_HAL\_SD\_SDIO\_DISABLE**  
**Return value:**None
- **\_\_HAL\_SD\_SDIO\_DMA\_ENABLE**  
**Return value:**None
- **\_\_HAL\_SD\_SDIO\_DMA\_DISABLE**  
**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\_RXOVERRR: 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:**None:

- **\_HAL\_SD\_SDIO\_DISABLE\_IT**

**Description:** 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\_RXACT: Data transmit in progress interrupt SDIO\_IT\_TXACT: 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:**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\_RXACT: Data transmit in progress SDIO\_FLAG\_TXACT: 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 Exported Types***

- **SD\_InitTypeDef**
- **SD\_TypeDef**

#### ***SD Private Define***

- **SDIO\_STATIC\_FLAGS**
- **SDIO\_CMD0TIMEOUT**
- **SD\_OCR\_ADDR\_OUT\_OF\_RANGE**
- **SD\_OCR\_ADDR\_MISALIGNED**
- **SD\_OCR\_BLOCK\_LEN\_ERR**
- **SD\_OCR\_ERASE\_SEQ\_ERR**
- **SD\_OCR\_BAD\_ERASE\_PARAM**
- **SD\_OCR\_WRITE\_PROT\_VIOLATION**
- **SD\_OCR\_LOCK\_UNLOCK\_FAILED**
- **SD\_OCR\_COM\_CRC\_FAILED**
- **SD\_OCR\_ILLEGAL\_CMD**
- **SD\_OCR\_CARD\_ECC\_FAILED**
- **SD\_OCR\_CC\_ERROR**
- **SD\_OCR\_GENERAL\_UNKNOWN\_ERROR**
- **SD\_OCR\_STREAM\_READ\_UNDERRUN**
- **SD\_OCR\_STREAM\_WRITE\_OVERRUN**
- **SD\_OCR\_CID\_CSD\_OVERWRIETE**
- **SD\_OCR\_WP\_ERASE\_SKIP**
- **SD\_OCR\_CARD\_ECC\_DISABLED**
- **SD\_OCR\_ERASE\_RESET**
- **SD\_OCR\_AKE\_SEQ\_ERROR**
- **SD\_OCR\_ERRORBITS**
- **SD\_R6\_GENERAL\_UNKNOWN\_ERROR**
- **SD\_R6\_ILLEGAL\_CMD**
- **SD\_R6\_COM\_CRC\_FAILED**
- **SD\_VOLTAGE\_WINDOW\_SD**
- **SD\_HIGH\_CAPACITY**
- **SD\_STD\_CAPACITY**
- **SD\_CHECK\_PATTERN**
- **SD\_MAX\_VOLT\_TRIAL**
- **SD\_ALLZERO**
- **SD\_WIDE\_BUS\_SUPPORT**
- **SD\_SINGLE\_BUS\_SUPPORT**
- **SD\_CARD\_LOCKED**
- **SD\_DATATIMEOUT**
- **SD\_0TO7BITS**
- **SD\_8TO15BITS**
- **SD\_16TO23BITS**
- **SD\_24TO31BITS**
- **SD\_MAX\_DATA\_LENGTH**
- **SD\_HALFFIFO**

- **SD\_HALFFIFOBYTES**
- **SD\_CCCC\_LOCK\_UNLOCK**
- **SD\_CCCC\_WRITE\_PROT**
- **SD\_CCCC\_ERASE**
- **SD\_SDIO\_SEND\_IF\_COND**

SDIO\_APP\_CMD should be sent before sending these commands.

## 38 HAL SMARTCARD Generic Driver

### 38.1 SMARTCARD Firmware driver registers structures

#### 38.1.1 SMARTCARD\_InitTypeDef

*SMARTCARD\_InitTypeDef* is defined in the `stm32l1xx_hal_smartcard.h`

##### 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:
  - IntegerDivider = ((PCLKx) / (8 \* (hsc->Init.BaudRate)))
  - FractionalDivider = ((IntegerDivider - ((uint32\_t) IntegerDivider)) \* 8) + 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. This parameter must be a number between Min\_Data = 0 and Max\_Data = 255

- ***uint32\_t SMARTCARD\_InitTypeDef::GuardTime*** Specifies the SmartCard Guard Time This parameter must be a number between Min\_Data = 0 and Max\_Data = 255
- ***uint32\_t SMARTCARD\_InitTypeDef::NACKState*** Specifies the SmartCard NACK Transmission state This parameter can be a value of ***SMARTCARD\_NACK\_State***

### 38.1.2 SMARTCARD\_HandleTypeDef

***SMARTCARD\_HandleTypeDef*** is defined in the `stm32l1xx_hal_smartcard.h`

#### 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 HAL\_SMARTCARD\_ErrorTypeDef 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 HAL\_SMARTCARD\_ErrorTypeDef SMARTCARD\_HandleTypeDef::ErrorCode*** SmartCard Error code

## 38.2 SMARTCARD Firmware driver API description

The following section lists the various functions of the SMARTCARD library.

### 38.2.1 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 USARTx interface clock.
  - b. SMARTCARD pins configuration:
    - Enable the clock for the SMARTCARD GPIOs.
    - Configure these 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.
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



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

#### 38.2.2

#### 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 (see below table)
  - 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

**Table 23: Frame formats**

| M bit | PCE bit | Smartcard frame            |
|-------|---------|----------------------------|
| 1     | 1       | SB   8 bit data   PB   STB |

Please refer to the ISO 7816-3 specification for more details. -@- 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)).

- [\*\*HAL\\_SMARTCARD\\_Init\(\)\*\*](#)
- [\*\*HAL\\_SMARTCARD\\_DelInit\(\)\*\*](#)
- [\*\*HAL\\_SMARTCARD\\_MspInit\(\)\*\*](#)
- [\*\*HAL\\_SMARTCARD\\_MspDelInit\(\)\*\*](#)

### 38.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the SMARTCARD data transfers.

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. 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.

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 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
2. Blocking mode APIs are :
    - HAL\_SMARTCARD\_Transmit()
    - HAL\_SMARTCARD\_Receive()
  3. Non Blocking mode APIs with Interrupt are :
    - HAL\_SMARTCARD\_Transmit\_IT()
    - HAL\_SMARTCARD\_Receive\_IT()
    - HAL\_SMARTCARD\_IRQHandler()
  4. Non Blocking mode functions with DMA are :
    - HAL\_SMARTCARD\_Transmit\_DMA()
    - HAL\_SMARTCARD\_Receive\_DMA()
  5. A set of Transfer Complete Callbacks are provided in non Blocking mode:
    - HAL\_SMARTCARD\_TxCpltCallback()
    - HAL\_SMARTCARD\_RxCpltCallback()
    - HAL\_SMARTCARD\_ErrorCallback()

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. 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. (#) 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 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 (#) Blocking mode APIs are :
- HAL\_SMARTCARD\_Transmit()
  - HAL\_SMARTCARD\_Receive() (#) Non Blocking mode APIs with Interrupt are :
  - HAL\_SMARTCARD\_Transmit\_IT()
  - HAL\_SMARTCARD\_Receive\_IT()
  - HAL\_SMARTCARD\_IRQHandler() (#) Non Blocking mode functions with DMA are :
  - HAL\_SMARTCARD\_Transmit\_DMA()
  - HAL\_SMARTCARD\_Receive\_DMA() (#) A set of Transfer Complete Callbacks are provided in non Blocking mode:
  - HAL\_SMARTCARD\_TxCpltCallback()
  - HAL\_SMARTCARD\_RxCpltCallback()

- HAL\_SMARTCARD\_ErrorCallback()
- [\*\*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\(\)\*\*](#)

### 38.2.4 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.
- [\*\*HAL\\_SMARTCARD\\_GetState\(\)\*\*](#)
- [\*\*HAL\\_SMARTCARD\\_GetError\(\)\*\*](#)

### 38.2.5 HAL\_SMARTCARD\_Init

|                      |                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SMARTCARD_Init (</b><br><a href="#"><b>SMARTCARD_HandleTypeDef * hsc</b></a> )                                                                                         |
| 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"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                       |

### 38.2.6 HAL\_SMARTCARD\_DelInit

|               |                                                  |
|---------------|--------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_SMARTCARD_DelInit (</b> |
|---------------|--------------------------------------------------|

***SMARTCARD\_HandleTypeDef \* hsc)***

|                      |                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Deinitializes the SMARTCARD peripheral.                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                           |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |

### 38.2.7 HAL\_SMARTCARD\_MspInit

|                      |                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SMARTCARD_MspInit (</b><br><b><i>SMARTCARD_HandleTypeDef * hsc)</i></b>                                                                                                           |
| Function Description | SMARTCARD MSP Init.                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |

### 38.2.8 HAL\_SMARTCARD\_MspDelInit

|                      |                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SMARTCARD_MspDelInit (</b><br><b><i>SMARTCARD_HandleTypeDef * hsc)</i></b>                                                                                                        |
| Function Description | SMARTCARD MSP DelInit.                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |

### 38.2.9 HAL\_SMARTCARD\_Transmit

|                      |                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SMARTCARD_Transmit (</code><br><code>SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t</code><br><code>Size, uint32_t Timeout)</code>                                                                                                                                                                              |
| Function Description | Sends an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> <li>• <b>Timeout</b> : Specify timeout value</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                             |

### 38.2.10 HAL\_SMARTCARD\_Receive

|                      |                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SMARTCARD_Receive (</code><br><code>SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t</code><br><code>Size, uint32_t Timeout)</code>                                                                                                                                                                                   |
| Function Description | Receive an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> <li>• <b>Timeout</b> : Specify timeout value</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                 |

### 38.2.11 HAL\_SMARTCARD\_Transmit\_IT

|               |                                                                                                                                     |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_SMARTCARD_Transmit_IT (</code><br><code>SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t</code> |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <b>Size)</b>                                                                                                                                                                                                                                                                                        |
| Function Description | Sends an amount of data in non-blocking mode.                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                 |
| Notes                | • None.                                                                                                                                                                                                                                                                                             |

### 38.2.12 HAL\_SMARTCARD\_Receive\_IT

|                      |                                                                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SMARTCARD_Receive_IT (</b><br><b>SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t</b><br><b>Size)</b>                                                                                                                                                                  |
| Function Description | Receives an amount of data in non-blocking mode.                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                     |
| Notes                | • None.                                                                                                                                                                                                                                                                                                 |

### 38.2.13 HAL\_SMARTCARD\_Transmit\_DMA

|                      |                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SMARTCARD_Transmit_DMA (</b><br><b>SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t</b><br><b>Size)</b>                                                                                                                                                            |
| Function Description | Sends an amount of data in non-blocking mode.                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |

---

|               |                                                                     |
|---------------|---------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul> |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>             |

### 38.2.14 HAL\_SMARTCARD\_Receive\_DMA

|                      |                                                                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SMARTCARD_Receive_DMA (</b><br><b>SMARTCARD_HandleTypeDef * hsc, uint8_t * pData, uint16_t</b><br><b>Size)</b>                                                                                                                                                                 |
| Function Description | Receive an amount of data in non-blocking mode.                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> </ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                     |
| Notes                | <ul style="list-style-type: none"><li>• When the SMARTCARD parity is enabled (PCE = 1) the data received contain the parity bit.</li></ul>                                                                                                                                                              |

### 38.2.15 HAL\_SMARTCARD\_IRQHandler

|                      |                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SMARTCARD_IRQHandler (</b><br><b>SMARTCARD_HandleTypeDef * hsc)</b>                                                                                                                 |
| Function Description | This function handles SMARTCARD interrupt request.                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                         |

### 38.2.16 HAL\_SMARTCARD\_TxCpltCallback

|                      |                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SMARTCARD_TxCpltCallback (</b><br><b>SMARTCARD_HandleTypeDef * hsc)</b>                                                                                                           |
| Function Description | Tx Transfer completed callbacks.                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |

### 38.2.17 HAL\_SMARTCARD\_RxCpltCallback

|                      |                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SMARTCARD_RxCpltCallback (</b><br><b>SMARTCARD_HandleTypeDef * hsc)</b>                                                                                                           |
| Function Description | Rx Transfer completed callbacks.                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |

### 38.2.18 HAL\_SMARTCARD\_ErrorCallback

|                      |                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SMARTCARD_ErrorCallback (</b><br><b>SMARTCARD_HandleTypeDef * hsc)</b>                                                                                                            |
| Function Description | SMARTCARD error callbacks.                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                       |

**38.2.19 HAL\_SMARTCARD\_GetState**

|                      |                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SMARTCARD_StateTypeDef</b><br><b>HAL_SMARTCARD_GetState ( <i>SMARTCARD_HandleTypeDef</i> * hsc)</b>                                                                                      |
| Function Description | Returns the SMARTCARD state.                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                       |

**38.2.20 HAL\_SMARTCARD\_GetError**

|                      |                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_SMARTCARD_GetError ( <i>SMARTCARD_HandleTypeDef</i> * hsc)</b>                                                                                                                  |
| Function Description | Return the SMARTCARD error code.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsc</b> : Pointer to a SMARTCARD_HandleTypeDef structure that contains the configuration information for the specified SMARTCARD module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>SMARTCARD Error Code</b></li> </ul>                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                       |

**38.3 SMARTCARD Firmware driver defines****38.3.1 SMARTCARD**

SMARTCARD

***SMARTCARD Clock Phase***

- **SMARTCARD\_PHASE\_1EDGE**
- **SMARTCARD\_PHASE\_2EDGE**
- **IS\_SMARTCARD\_PHASE**

***SMARTCARD Clock Polarity***

- **SMARTCARD\_POLARITY\_LOW**
- **SMARTCARD\_POLARITY\_HIGH**
- **IS\_SMARTCARD\_POLARITY**

**SMARTCARD DMA requests**

- **SMARTCARD\_DMAREQ\_TX**
- **SMARTCARD\_DMAREQ\_RX**

**SMARTCARD Exported Macros**

- **\_\_HAL\_SMARTCARD\_RESET\_HANDLE\_STATE**

**Description:** Reset SMARTCARD handle state.

**Parameters:** `__HANDLE__`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3.

**Return value:**None

- **\_\_HAL\_SMARTCARD\_FLUSH\_DRREGISTER**

**Description:** Flushes the Smartcard DR register.

**Parameters:** `__HANDLE__`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3.

- **\_\_HAL\_SMARTCARD\_GET\_FLAG**

**Description:** Checks whether the specified Smartcard flag is set or not.

**Parameters:** `__HANDLE__`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3. `__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:** Clears the specified Smartcard pending flags.

**Parameters:** `__HANDLE__`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3. `__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

- **\_\_HAL\_SMARTCARD\_CLEAR\_PEFLAG**

**Description:** Clear the SMARTCARD PE pending flag.

**Parameters:** `__HANDLE__`: specifies the USART Handle. This parameter can be USARTx with x: 1, 2 or 3.

**Return value:**None

- **\_\_HAL\_SMARTCARD\_CLEAR\_FEFLAG**

**Description:** Clear the SMARTCARD FE pending flag.

**Parameters:** `__HANDLE__`: specifies the USART Handle. This parameter can be USARTx with x: 1, 2 or 3.

**Return value:**None

- **\_\_HAL\_SMARTCARD\_CLEAR\_NEFLAG**

**Description:** Clear the SMARTCARD NE pending flag.

**Parameters:** `__HANDLE__`: specifies the USART Handle. This parameter can be USARTx with x: 1, 2 or 3.

**Return value:**None

- **\_\_HAL\_SMARTCARD\_CLEAR\_OREFLAG**

**Description:** Clear the SMARTCARD ORE pending flag.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx with x: 1, 2 or 3.

**Return value:**None:

- `_HAL_SMARTCARD_CLEAR_IDLEFLAG`

**Description:** Clear the SMARTCARD IDLE pending flag.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx with x: 1, 2 or 3.

**Return value:**None:

- `_HAL_SMARTCARD_ENABLE_IT`

**Description:** Enables the specified SmartCard interrupt.

**Parameters:** `_HANDLE_`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3. `_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)

- `_HAL_SMARTCARD_DISABLE_IT`

**Description:** Disables the specified SmartCard interrupts.

**Parameters:** `_HANDLE_`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3. `_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:** Checks whether the specified SmartCard interrupt has occurred or not.

**Parameters:** `_HANDLE_`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3. `_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).

- `_HAL_SMARTCARD_ENABLE`

**Description:** Enable the USART associated to the SMARTCARD Handle.

**Parameters:** `_HANDLE_`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3.

**Return value:**None:

- `_HAL_SMARTCARD_DISABLE`

**Description:** Disable the USART associated to the SMARTCARD Handle.

**Parameters:** `_HANDLE_`: specifies the SMARTCARD Handle. This parameter can be USARTx with x: 1, 2 or 3.

**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 interruptions flag mask***

- **SMARTCARD\_IT\_MASK**

#### ***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**
- **IS\_SMARTCARD\_LASTBIT**

#### ***SMARTCARD Mode***

- **SMARTCARD\_MODE\_RX**
- **SMARTCARD\_MODE\_TX**
- **SMARTCARD\_MODE\_TX\_RX**
- **IS\_SMARTCARD\_MODE**

#### ***SMARTCARD NACK State***

- **SMARTCARD\_NACK\_ENABLED**
- **SMARTCARD\_NACK\_DISABLED**
- **IS\_SMARTCARD\_NACK\_STATE**

#### ***SMARTCARD Parity***

- **SMARTCARD\_PARITY\_NONE**
- **SMARTCARD\_PARITY\_EVEN**
- **SMARTCARD\_PARITY\_ODD**
- **IS\_SMARTCARD\_PARITY**

#### ***SMARTCARD Private Constants***

- **SMARTCARD\_TIMEOUT\_VALUE**

#### ***SMARTCARD Private Macros***

- **SMARTCARD\_DMA\_REQUEST\_ENABLE**

**Description:** Macro to enable the SmartCard DMA request.

**Parameters:** `HANDLE`: specifies the SmartCard Handle. `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

- **SMARTCARD\_DMA\_REQUEST\_DISABLE**

**Description:** Macro to disable the SmartCard DMA request.

**Parameters:** `HANDLE`: specifies the SmartCard Handle. `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

- **SMARTCARD\_DIV**

- **SMARTCARD\_DIVMANT**
- **SMARTCARD\_DIVFRAQ**
- **SMARTCARD\_BRR**
- **IS\_SMARTCARD\_BAUDRATE**

**Description:** Check the Baud rate range.

**Parameters:** BAUDRATE: Baud rate set by the configuration function.

**Return value:** Test: result (TRUE or FALSE)

***SMARTCARD Number of Stop Bits***

- **SMARTCARD\_STOPBITS\_1**
- **SMARTCARD\_STOPBITS\_0\_5**
- **SMARTCARD\_STOPBITS\_2**
- **SMARTCARD\_STOPBITS\_1\_5**
- **IS\_SMARTCARD\_STOPBITS**

***SMARTCARD Word Length***

- **SMARTCARD\_WORDLENGTH\_8B**
- **SMARTCARD\_WORDLENGTH\_9B**
- **IS\_SMARTCARD\_WORD\_LENGTH**

## 39 HAL SPI Generic Driver

### 39.1 SPI Firmware driver registers structures

#### 39.1.1 SPI\_InitTypeDef

`SPI_InitTypeDef` is defined in the `stm32l1xx_hal_spi.h`

##### 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

### 39.1.2 SPI\_HandleTypeDef

SPI\_HandleTypeDef is defined in the `stm32l1xx_hal_spi.h`

#### Data Fields

- `SPI_TypeDef * Instance`
- `SPI_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`
- `void(* RxISR`
- `void(* TxISR`
- `HAL_LockTypeDef Lock`
- `__IO HAL_SPI_StateTypeDef State`
- `__IO HAL_SPI_ErrorTypeDef ErrorCode`

#### Field Documentation

- `SPI_TypeDef* __SPI_HandleTypeDef::Instance`
- `SPI_InitTypeDef __SPI_HandleTypeDef::Init`
- `uint8_t* __SPI_HandleTypeDef::pTxBuffPtr`
- `uint16_t __SPI_HandleTypeDef::TxXferSize`
- `uint16_t __SPI_HandleTypeDef::TxXferCount`
- `uint8_t* __SPI_HandleTypeDef::pRxBuffPtr`
- `uint16_t __SPI_HandleTypeDef::RxXferSize`
- `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 HAL_SPI_ErrorTypeDef __SPI_HandleTypeDef::ErrorCode`

## 39.2 SPI Firmware driver API description

The following section lists the various functions of the SPI library.

### 39.2.1 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

### 39.2.2 Initialization and de-initialization functions

This subsection provides a set of functions allowing to initialize and de-initialize 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\_DeInit() to restore the default configuration of the selected SPIx peripheral.
- **HAL\_SPI\_Init()**
- **HAL\_SPI\_DeInit()**

- [\*HAL\\_SPI\\_MspInit\(\)\*](#)
- [\*HAL\\_SPI\\_MspDeInit\(\)\*](#)

### 39.2.3 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\_TxRx\_CpltCallback()* 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. Blocking mode APIs are :
  - *HAL\_SPI\_Transmit()* in 1Line (simplex) and 2Lines (full duplex) mode
  - *HAL\_SPI\_Receive()* in 1Line (simplex) and 2Lines (full duplex) mode
  - *HAL\_SPI\_TransmitReceive()* in full duplex mode
3. Non Blocking mode API's with Interrupt are :
  - *HAL\_SPI\_Transmit\_IT()* in 1Line (simplex) and 2Lines (full duplex) mode
  - *HAL\_SPI\_Receive\_IT()* in 1Line (simplex) and 2Lines (full duplex) mode
  - *HAL\_SPI\_TransmitReceive\_IT()* in full duplex mode
  - *HAL\_SPI\_IRQHandler()*
4. Non Blocking mode functions with DMA are :
  - *HAL\_SPI\_Transmit\_DMA()* in 1Line (simplex) and 2Lines (full duplex) mode
  - *HAL\_SPI\_Receive\_DMA()* in 1Line (simplex) and 2Lines (full duplex) mode
  - *HAL\_SPI\_TransmitReceive\_DMA()* in full duplex mode
5. A set of Transfer Complete Callbacks are provided in non Blocking mode:
  - *HAL\_SPI\_TxCpltCallback()*
  - *HAL\_SPI\_RxCpltCallback()*
  - *HAL\_SPI\_TxRx\_CpltCallback()*
  - *HAL\_SPI\_TxHalfCpltCallback()*
  - *HAL\_SPI\_RxHalfCpltCallback()*
  - *HAL\_SPI\_TxRxHalfCpltCallback()*
  - *HAL\_SPI\_ErrorCallback()*
  - [\*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()`

### 39.2.4 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
- `HAL_SPI_GetState()`
- `HAL_SPI_GetError()`

### 39.2.5 HAL\_SPI\_Init

|                      |                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SPI_Init ( SPI_HandleTypeDef * hspi)</code>                                                                                                                   |
| Function Description | Initializes the SPI according to the specified parameters in the <code>SPI_InitTypeDef</code> and create the associated handle.                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <code>hspi</code> : pointer to a <code>SPI_HandleTypeDef</code> structure that contains the configuration information for SPI module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                 |

### 39.2.6 HAL\_SPI\_DeInit

|                      |                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SPI_DeInit ( SPI_HandleTypeDef * hspi)</code>                                                                                                                 |
| Function Description | Deinitializes the SPI peripheral.                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <code>hspi</code> : pointer to a <code>SPI_HandleTypeDef</code> structure that contains the configuration information for SPI module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                 |

### 39.2.7 HAL\_SPI\_MspInit

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_MspInit ( SPI_HandleTypeDef * hspi)</b>                                                                                                              |
| Function Description | SPI MSP Init.                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.8 HAL\_SPI\_MspDelInit

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_MspDelInit ( SPI_HandleTypeDef * hspi)</b>                                                                                                           |
| Function Description | SPI MSP DelInit.                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.9 HAL\_SPI\_Transmit

|                      |                                                                                                                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_Transmit ( SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                            |
| Function Description | Transmit an amount of data in blocking mode.                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li><li>• <b>pData</b> : pointer to data buffer</li><li>• <b>Size</b> : amount of data to be sent</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |

---

|               |                                                                     |
|---------------|---------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul> |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>             |

### 39.2.10 HAL\_SPI\_Receive

|                      |                                                                                                                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_Receive ( SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                             |
| Function Description | Receive an amount of data in blocking mode.                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li><li>• <b>pData</b> : pointer to data buffer</li><li>• <b>Size</b> : amount of data to be sent</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                            |

### 39.2.11 HAL\_SPI\_TransmitReceive

|                      |                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_TransmitReceive ( SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                                                                                 |
| Function Description | Transmit and Receive an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li><li>• <b>pTxData</b> : pointer to transmission data buffer</li><li>• <b>pRxData</b> : pointer to reception data buffer to be</li><li>• <b>Size</b> : amount of data to be sent</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                                                             |

### 39.2.12 HAL\_SPI\_Transmit\_IT

|                      |                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_Transmit_IT (</b><br><b>SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                         |
| Function Description | Transmit an amount of data in no-blocking mode with Interrupt.                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li> <li>• <b>pData</b> : pointer to data buffer</li> <li>• <b>Size</b> : amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                  |

### 39.2.13 HAL\_SPI\_Receive\_IT

|                      |                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_Receive_IT (</b><br><b>SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                          |
| Function Description | Receive an amount of data in no-blocking mode with Interrupt.                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li> <li>• <b>pData</b> : pointer to data buffer</li> <li>• <b>Size</b> : amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                  |

### 39.2.14 HAL\_SPI\_TransmitReceive\_IT

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_TransmitReceive_IT (</b><br><b>SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)</b>                         |
| Function Description | Transmit and Receive an amount of data in no-blocking mode with Interrupt.                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li> </ul> |

- |               |                                                                                                                                                                                                                                |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"> <li>• <b>pTxData</b> : pointer to transmission data buffer</li> <li>• <b>pRxData</b> : pointer to reception data buffer to be</li> <li>• <b>Size</b> : amount of data to be sent</li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> <li>• None.</li> </ul>                                                                                                                                         |

### 39.2.15 HAL\_SPI\_Transmit\_DMA

|                      |                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_Transmit_DMA (</b><br><b>SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                        |
| Function Description | Transmit an amount of data in no-blocking mode with DMA.                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li> <li>• <b>pData</b> : pointer to data buffer</li> <li>• <b>Size</b> : amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                  |

### 39.2.16 HAL\_SPI\_Receive\_DMA

|                      |                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_Receive_DMA (</b><br><b>SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)</b>                                                                                                      |
| Function Description | Receive an amount of data in no-blocking mode with DMA.                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li> <li>• <b>pData</b> : pointer to data buffer</li> </ul> |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Size</b> : amount of data to be sent</li> </ul>                                                                                                                             |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>• When the CRC feature is enabled the pData Length must be Size + 1.</li> </ul>                                                                                                  |

### 39.2.17 HAL\_SPI\_TransmitReceive\_DMA

|                      |                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SPI_TransmitReceive_DMA (SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)</code>                                                                                                                                                          |
| Function Description | Transmit and Receive an amount of data in no-blocking mode with DMA.                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li> <li>• <b>pTxData</b> : pointer to transmission data buffer</li> <li>• <b>pRxData</b> : pointer to reception data buffer</li> </ul> |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>Size</b> : amount of data to be sent</li> </ul>                                                                                                                                                                                                         |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• When the CRC feature is enabled the pRxData Length must be Size + 1</li> </ul>                                                                                                                                                                             |

### 39.2.18 HAL\_SPI\_DMAPause

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SPI_DMAPause (SPI_HandleTypeDef * hspi)</code>                                                                                                           |
| Function Description | Pauses the DMA Transfer.                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for the specified SPI module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                |

Notes      • None.

### 39.2.19 HAL\_SPI\_DMAResume

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SPI_DMAResume (SPI_HandleTypeDef * hspi)</code> |
| Function Description | Resumes the DMA Transfer.                                                   |

|               |                                                                                                                                                                                    |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for the specified SPI module.</li></ul> |
| Return values | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                            |

### 39.2.20 HAL\_SPI\_DMASStop

|                      |                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SPI_DMASStop ( SPI_HandleTypeDef * hspi)</b>                                                                                                               |
| Function Description | Stops the DMA Transfer.                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                             |

### 39.2.21 HAL\_SPI\_IRQHandler

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_IRQHandler ( SPI_HandleTypeDef * hspi)</b>                                                                                                           |
| Function Description | This function handles SPI interrupt request.                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.22 HAL\_SPI\_TxCpltCallback

---

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_TxCpltCallback ( SPI_HandleTypeDef * hspi)</b>                                                                                                       |
| Function Description | Tx Transfer completed callbacks.                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.23 HAL\_SPI\_RxCpltCallback

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_RxCpltCallback ( SPI_HandleTypeDef * hspi)</b>                                                                                                       |
| Function Description | Rx Transfer completed callbacks.                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.24 HAL\_SPI\_TxRxCpltCallback

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_TxRxCpltCallback ( SPI_HandleTypeDef * hspi)</b>                                                                                                     |
| Function Description | Tx and Rx Transfer completed callbacks.                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.25 HAL\_SPI\_TxHalfCpltCallback

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_TxHalfCpltCallback ( SPI_HandleTypeDef * hspi)</b>                                                                                                   |
| Function Description | Tx Half Transfer completed callbacks.                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.26 HAL\_SPI\_RxHalfCpltCallback

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_RxHalfCpltCallback ( SPI_HandleTypeDef * hspi)</b>                                                                                                   |
| Function Description | Rx Half Transfer completed callbacks.                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.27 HAL\_SPI\_TxRxHalfCpltCallback

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_TxRxHalfCpltCallback ( SPI_HandleTypeDef * hspi)</b>                                                                                                 |
| Function Description | Tx and Rx Transfer completed callbacks.                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.28 HAL\_SPI\_ErrorCallback

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SPI_ErrorCallback ( SPI_HandleTypeDef * hspi)</b>                                                                                                        |
| Function Description | SPI error callbacks.                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                              |

### 39.2.29 HAL\_SPI\_GetState

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SPI_StateTypeDef HAL_SPI_GetState ( SPI_HandleTypeDef * hspi)</b>                                                                                             |
| Function Description | Return the SPI state.                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                                                                                   |

### 39.2.30 HAL\_SPI\_GetError

|                      |                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_SPI_ErrorTypeDef HAL_SPI_GetError ( SPI_HandleTypeDef * hspi)</b>                                                                                             |
| Function Description | Return the SPI error code.                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hspi</b> : pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>SPI Error Code</b></li></ul>                                                                                              |

## 39.3 SPI Firmware driver defines

### 39.3.1 SPI

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\_DISABLED
- SPI\_CRCCALCULATION\_ENABLED
- 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 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. `_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 Private Constants**

- **SPI\_TIMEOUT\_VALUE**

#### **SPI Private Macros**

- **SPI\_1LINE\_TX**  
**Description:** Sets the SPI transmit-only mode.  
**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\_1LINE\_RX**  
**Description:** Sets the SPI receive-only mode.  
**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\_RESET\_CRC**  
**Description:** Resets the CRC calculation of 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 Slave Select management**

- **SPI\_NSS\_SOFT**
- **SPI\_NSS\_HARD\_INPUT**
- **SPI\_NSS\_HARD\_OUTPUT**
- **IS\_SPI\_NSS**

#### **SPI TI mode**

- **SPI\_TIMODE\_DISABLED**
- **SPI\_TIMODE\_ENABLED**
- **IS\_SPI\_TIMODE**

## 40 HAL SRAM Generic Driver

### 40.1 SRAM Firmware driver registers structures

#### 40.1.1 SRAM\_HandleTypeDef

*SRAM\_HandleTypeDef* is defined in the `stm32l1xx_hal_sram.h`

##### 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

### 40.2 SRAM Firmware driver API description

The following section lists the various functions of the SRAM library.

#### 40.2.1 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 hsram;` 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 \_\_FSMC\_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()

#### 40.2.2 SRAM Initialization and de\_initialization functions

This section provides functions allowing to initialize/de-initialize the SRAM memory

- [\*\*HAL\\_SRAM\\_Init\(\)\*\*](#)
- [\*\*HAL\\_SRAM\\_DeInit\(\)\*\*](#)
- [\*\*HAL\\_SRAM\\_MspInit\(\)\*\*](#)
- [\*\*HAL\\_SRAM\\_MspDeInit\(\)\*\*](#)
- [\*\*HAL\\_SRAM\\_DMA\\_XferCpltCallback\(\)\*\*](#)
- [\*\*HAL\\_SRAM\\_DMA\\_XferErrorCallback\(\)\*\*](#)

#### 40.2.3 SRAM Input and Output functions

This section provides functions allowing to use and control the SRAM memory

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

#### 40.2.4 SRAM Control functions

This subsection provides a set of functions allowing to control dynamically the SRAM interface.

- *HAL\_SRAM\_WriteOperation\_Enable()*
- *HAL\_SRAM\_WriteOperation\_Disable()*

#### 40.2.5 SRAM State functions

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

- *HAL\_SRAM\_GetState()*

#### 40.2.6 HAL\_SRAM\_Init

|                      |                                                                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SRAM_Init ( <i>SRAM_HandleTypeDef</i> * hsram, <i>FSMC_NORSRAM_TimingTypeDef</i> * Timing, <i>FSMC_NORSRAM_TimingTypeDef</i> * ExtTiming)</b>                                                                                                                                                     |
| Function Description | Performs the SRAM device initialization sequence.                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>Timing</b> : Pointer to SRAM control timing structure</li> <li>• <b>ExtTiming</b> : Pointer to SRAM extended mode timing structure</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                        |
| Notes                | • None.                                                                                                                                                                                                                                                                                                                    |

#### 40.2.7 HAL\_SRAM\_DelInit

|                      |                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SRAM_DelInit ( <i>SRAM_HandleTypeDef</i> * hsram)</b>                                                                                            |
| Function Description | Performs the SRAM device De-initialization sequence.                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                       |
| Notes                | • None.                                                                                                                                                                   |

#### 40.2.8 HAL\_SRAM\_MspInit

|                      |                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SRAM_MspInit ( <i>SRAM_HandleTypeDef</i> * hsram)</b>                                                                                                       |
| Function Description | SRAM MSP Init.                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                 |

#### 40.2.9 HAL\_SRAM\_MspDeInit

|                      |                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SRAM_MspDeInit ( <i>SRAM_HandleTypeDef</i> * hsram)</b>                                                                                                     |
| Function Description | SRAM MSP DeInit.                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                 |

#### 40.2.10 HAL\_SRAM\_DMA\_XferCpltCallback

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SRAM_DMA_XferCpltCallback ( <i>DMA_HandleTypeDef</i> * hdma)</b>                                                                                           |
| Function Description | DMA transfer complete callback.                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hdma</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                |

#### 40.2.11 HAL\_SRAM\_DMA\_XferErrorCallback

|                      |                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_SRAM_DMA_XferErrorCallback (</b><br><b>DMA_HandleTypeDef * hdma)</b>                                                                                         |
| Function Description | DMA transfer complete error callback.                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hdma</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                |

#### 40.2.12 HAL\_SRAM\_Read\_8b

|                      |                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SRAM_Read_8b (</b><br><b>SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint8_t * pDstBuffer, uint32_t BufferSize)</b>                                                                                                                                                                                                                       |
| Function Description | Reads 8-bit buffer from SRAM memory.                                                                                                                                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>pAddress</b> : Pointer to read start address</li> <li>• <b>pDstBuffer</b> : Pointer to destination buffer</li> <li>• <b>BufferSize</b> : Size of the buffer to read from memory</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                               |

#### 40.2.13 HAL\_SRAM\_Write\_8b

|                      |                                                                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SRAM_Write_8b (</b><br><b>SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint8_t * pSrcBuffer, uint32_t BufferSize)</b>                                                                                    |
| Function Description | Writes 8-bit buffer to SRAM memory.                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>pAddress</b> : Pointer to write start address</li> </ul> |

- **pSrcBuffer** : Pointer to source buffer to write
  - **BufferSize** : Size of the buffer to write to memory
- Return values
- **HAL status**
- Notes
- None.

#### 40.2.14 HAL\_SRAM\_Read\_16b

|                      |                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SRAM_Read_16b (<br/>    <b>SRAM_HandleTypeDef</b> * hsram, uint32_t * pAddress, uint16_t<br/>    * pDstBuffer, uint32_t BufferSize)</code>                                                                                                                                                                                                  |
| Function Description | Reads 16-bit buffer from SRAM memory.                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>pAddress</b> : Pointer to read start address</li> <li>• <b>pDstBuffer</b> : Pointer to destination buffer</li> <li>• <b>BufferSize</b> : Size of the buffer to read from memory</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                     |
| Notes                | • None.                                                                                                                                                                                                                                                                                                                                                                 |

#### 40.2.15 HAL\_SRAM\_Write\_16b

|                      |                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SRAM_Write_16b (<br/>    <b>SRAM_HandleTypeDef</b> * hsram, uint32_t * pAddress, uint16_t<br/>    * pSrcBuffer, uint32_t BufferSize)</code>                                                                                                                                                                                                     |
| Function Description | Writes 16-bit buffer to SRAM memory.                                                                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>pAddress</b> : Pointer to write start address</li> <li>• <b>pSrcBuffer</b> : Pointer to source buffer to write</li> <li>• <b>BufferSize</b> : Size of the buffer to write to memory</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                         |
| Notes                | • None.                                                                                                                                                                                                                                                                                                                                                                     |

#### 40.2.16 HAL\_SRAM\_Read\_32b

|                      |                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SRAM_Read_32b (</code><br><code>SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint32_t</code><br><code>* pDstBuffer, uint32_t BufferSize)</code>                                                                                                                                                                                         |
| Function Description | Reads 32-bit buffer from SRAM memory.                                                                                                                                                                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>pAddress</b> : Pointer to read start address</li> <li>• <b>pDstBuffer</b> : Pointer to destination buffer</li> <li>• <b>BufferSize</b> : Size of the buffer to read from memory</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                               |

#### 40.2.17 HAL\_SRAM\_Write\_32b

|                      |                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SRAM_Write_32b (</code><br><code>SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint32_t</code><br><code>* pSrcBuffer, uint32_t BufferSize)</code>                                                                                                                                                                                            |
| Function Description | Writes 32-bit buffer to SRAM memory.                                                                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>pAddress</b> : Pointer to write start address</li> <li>• <b>pSrcBuffer</b> : Pointer to source buffer to write</li> <li>• <b>BufferSize</b> : Size of the buffer to write to memory</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                   |

#### 40.2.18 HAL\_SRAM\_Read\_DMA

|               |                                                                                                                              |
|---------------|------------------------------------------------------------------------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef HAL_SRAM_Read_DMA (</code><br><code>SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint32_t</code> |
|---------------|------------------------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <b>* pDstBuffer, uint32_t BufferSize)</b>                                                                                                                                                                                                                                                                                                                               |
| Function Description | Reads a Words data from the SRAM memory using DMA transfer.                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>pAddress</b> : Pointer to read start address</li> <li>• <b>pDstBuffer</b> : Pointer to destination buffer</li> <li>• <b>BufferSize</b> : Size of the buffer to read from memory</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                     |
| Notes                | • None.                                                                                                                                                                                                                                                                                                                                                                 |

#### 40.2.19 HAL\_SRAM\_Write\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SRAM_Write_DMA (</b><br><b>SRAM_HandleTypeDef * hsram, uint32_t * pAddress, uint32_t</b><br><b>* pSrcBuffer, uint32_t BufferSize)</b>                                                                                                                                                                                                              |
| Function Description | Writes a Words data buffer to SRAM memory using DMA transfer.                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> <li>• <b>pAddress</b> : Pointer to write start address</li> <li>• <b>pSrcBuffer</b> : Pointer to source buffer to write</li> <li>• <b>BufferSize</b> : Size of the buffer to write to memory</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                         |
| Notes                | • None.                                                                                                                                                                                                                                                                                                                                                                     |

#### 40.2.20 HAL\_SRAM\_WriteOperation\_Enable

|                      |                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_SRAM_WriteOperation_Enable (</b><br><b>SRAM_HandleTypeDef * hsram)</b>                                                                           |
| Function Description | Enables dynamically SRAM write operation.                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                       |
| Notes                | • None.                                                                                                                                                                   |

### 40.2.21 HAL\_SRAM\_WriteOperation\_Disable

|                      |                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_SRAM_WriteOperation_Disable (</code><br><code>SRAM_HandleTypeDef * hsram)</code>                                                            |
| Function Description | Disables dynamically SRAM write operation.                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                     |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                 |

### 40.2.22 HAL\_SRAM\_GetState

|                      |                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_SRAM_StateTypeDef HAL_SRAM_GetState (</code><br><code>SRAM_HandleTypeDef * hsram)</code>                                                                      |
| Function Description | Returns the SRAM controller state.                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hsram</b> : pointer to a SRAM_HandleTypeDef structure that contains the configuration information for SRAM module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                                                                                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                 |

## 40.3 SRAM Firmware driver defines

### 40.3.1 SRAM

SRAM

*SRAM Exported macro*

- `__HAL_SRAM_RESET_HANDLE_STATE`  
**Description:** Reset SRAM handle state.  
**Parameters:** `__HANDLE__`: SRAM handle  
**Return value:**None:

## 41 HAL TIM Generic Driver

### 41.1 TIM Firmware driver registers structures

#### 41.1.1 TIM\_Base\_InitTypeDef

*TIM\_Base\_InitTypeDef* is defined in the `stm32l1xx_hal_tim.h`

##### 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](#)

#### 41.1.2 TIM\_OC\_InitTypeDef

*TIM\_OC\_InitTypeDef* is defined in the `stm32l1xx_hal_tim.h`

##### 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\*\*](#).

#### 41.1.3 TIM\_OnePulse\_InitTypeDef

**TIM\_OnePulse\_InitTypeDef** is defined in the `stm32l1xx_hal_tim.h`

##### 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

#### 41.1.4 TIM\_IC\_InitTypeDef

**TIM\_IC\_InitTypeDef** is defined in the `stm32l1xx_hal_tim.h`

##### 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

#### 41.1.5 TIM\_Encoder\_InitTypeDef

*TIM\_Encoder\_InitTypeDef* is defined in the `stm32l1xx_hal_tim.h`

##### 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

#### 41.1.6 TIM\_ClockConfigTypeDef

*TIM\_ClockConfigTypeDef* is defined in the `stm32l1xx_hal_tim.h`

##### 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

### 41.1.7 TIM\_ClearInputConfigTypeDef

*TIM\_ClearInputConfigTypeDef* is defined in the `stm32l1xx_hal_tim.h`

#### 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

### 41.1.8 TIM\_SlaveConfigTypeDef

*TIM\_SlaveConfigTypeDef* is defined in the `stm32l1xx_hal_tim.h`

#### 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

#### 41.1.9 **TIM\_HandleTypeDef**

***TIM\_HandleTypeDef*** is defined in the `stm32l1xx_hal_tim.h`

##### 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 ***DMA\_HandleTypeDef***
- ***HAL\_LockTypeDef TIM\_HandleTypeDef::Lock*** Locking object
- ***\_\_IO HAL\_TIM\_StateTypeDef TIM\_HandleTypeDef::State*** TIM operation state

## 41.2 TIM Firmware driver API description

The following section lists the various functions of the TIM library.

### 41.2.1 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

#### 41.2.2 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 \_\_TIMx\_CLK\_ENABLE();
  - b. TIM pins configuration
    - Enable the clock for the TIM GPIOs using the following function: \_\_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()

### 41.2.3 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.
- [\*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\(\)\*](#)

### 41.2.4 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.
- [\*HAL\\_TIM\\_OC\\_Init\(\)\*](#)
- [\*HAL\\_TIM\\_OC\\_DelInit\(\)\*](#)
- [\*HAL\\_TIM\\_OC\\_MspInit\(\)\*](#)
- [\*HAL\\_TIM\\_OC\\_MspDelInit\(\)\*](#)
- [\*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\(\)\*](#)

### 41.2.5 Time PWM functions

This section provides functions allowing to:

- Initialize and configure the TIM OPWM.

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

#### 41.2.6 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.
- [\*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\(\)\*](#)

#### 41.2.7 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.
- [\*HAL\\_TIM\\_OnePulse\\_Init\(\)\*](#)
- [\*HAL\\_TIM\\_OnePulse\\_DelInit\(\)\*](#)
- [\*HAL\\_TIM\\_OnePulse\\_MspInit\(\)\*](#)
- [\*HAL\\_TIM\\_OnePulse\\_MspDelInit\(\)\*](#)
- [\*HAL\\_TIM\\_OnePulse\\_Start\(\)\*](#)
- [\*HAL\\_TIM\\_OnePulse\\_Stop\(\)\*](#)
- [\*HAL\\_TIM\\_OnePulse\\_Start\\_IT\(\)\*](#)
- [\*HAL\\_TIM\\_OnePulse\\_Stop\\_IT\(\)\*](#)

#### 41.2.8 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.
- [\*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\(\)\*](#)

#### 41.2.9 IRQ handler management

This section provides Timer IRQ handler function.

- [\*HAL\\_TIM\\_IRQHandler\(\)\*](#)

#### 41.2.10 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.
- [\*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_ReadCapturedValue()`

#### 41.2.11 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
- `HAL_TIM_PeriodElapsedCallback()`
- `HAL_TIM_OC_DelayElapsedCallback()`
- `HAL_TIM_IC_CaptureCallback()`
- `HAL_TIM_PWM_PulseFinishedCallback()`
- `HAL_TIM_TriggerCallback()`
- `HAL_TIM_ErrorCallback()`

#### 41.2.12 Peripheral State functions

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

- `HAL_TIM_Base_GetState()`
- `HAL_TIM_OC_GetState()`
- `HAL_TIM_PWM_GetState()`
- `HAL_TIM_IC_GetState()`
- `HAL_TIM_OnePulse_GetState()`
- `HAL_TIM_Encoder_GetState()`

#### 41.2.13 HAL\_TIM\_Base\_Init

|                      |                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_Base_Init (</code><br><code>TIM_HandleTypeDef * htim)</code>                                                     |
| Function Description | Initializes the TIM Time base Unit according to the specified parameters in the <code>TIM_HandleTypeDef</code> and create the associated handle. |
| Parameters           | <ul style="list-style-type: none"> <li>• <code>htim</code> : TIM Base handle</li> </ul>                                                          |
| Return values        | <ul style="list-style-type: none"> <li>• <code>HAL status</code></li> </ul>                                                                      |

## Notes

- None.

#### 41.2.14 HAL\_TIM\_Base\_DeInit

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Base_DeInit ( <i>TIM_HandleTypeDef</i> * htim)</b> |
| Function Description | DeInitializes the TIM Base peripheral.                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM Base handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                         |

#### 41.2.15 HAL\_TIM\_Base\_MspInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_Base_MspInit ( <i>TIM_HandleTypeDef</i> * htim)</b>        |
| Function Description | Initializes the TIM Base MSP.                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 41.2.16 HAL\_TIM\_Base\_MspDeInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_Base_MspDeInit ( <i>TIM_HandleTypeDef</i> * htim)</b>      |
| Function Description | DeInitializes TIM Base MSP.                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 41.2.17 HAL\_TIM\_Base\_Start

|                      |                                                                                          |
|----------------------|------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Base_Start (</b><br><b><i>TIM_HandleTypeDef * htim)</i></b> |
| Function Description | Starts the TIM Base generation.                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li></ul>             |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                  |

#### 41.2.18 HAL\_TIM\_Base\_Stop

|                      |                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Base_Stop (</b><br><b><i>TIM_HandleTypeDef * htim)</i></b> |
| Function Description | Stops the TIM Base generation.                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li></ul>            |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                     |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                 |

#### 41.2.19 HAL\_TIM\_Base\_Start\_IT

|                      |                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Base_Start_IT (</b><br><b><i>TIM_HandleTypeDef * htim)</i></b> |
| Function Description | Starts the TIM Base generation in interrupt mode.                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li></ul>                |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                         |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                     |

#### 41.2.20 HAL\_TIM\_Base\_Stop\_IT

|                      |                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Base_Stop_IT (</b><br><b><i>TIM_HandleTypeDef * htim)</i></b> |
| Function Description | Stops the TIM Base generation in interrupt mode.                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li></ul>               |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                    |

#### 41.2.21 HAL\_TIM\_Base\_Start\_DMA

|                      |                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Base_Start_DMA (</b><br><b><i>TIM_HandleTypeDef * htim, uint32_t * pData, uint16_t Length)</i></b>                                                                                            |
| Function Description | Starts the TIM Base generation in DMA mode.                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li><li>• <b>pData</b> : The source Buffer address.</li><li>• <b>Length</b> : The length of data to be transferred from memory to peripheral.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                    |

#### 41.2.22 HAL\_TIM\_Base\_Stop\_DMA

|                      |                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Base_Stop_DMA (</b><br><b><i>TIM_HandleTypeDef * htim)</i></b> |
| Function Description | Stops the TIM Base generation in DMA mode.                                                  |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li></ul>                |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                         |

## Notes

- None.

#### 41.2.23 HAL\_TIM\_OC\_Init

|                      |                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OC_Init ( <i>TIM_HandleTypeDef</i> * htim)</b>                                                                |
| Function Description | Initializes the TIM Output Compare according to the specified parameters in the <i>TIM_HandleTypeDef</i> and create the associated handle. |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM Output Compare handle</li></ul>                                                  |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                        |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                    |

#### 41.2.24 HAL\_TIM\_OC\_DeInit

|                      |                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OC_DeInit ( <i>TIM_HandleTypeDef</i> * htim)</b>             |
| Function Description | Deinitializes the TIM peripheral.                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM Output Compare handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                   |

#### 41.2.25 HAL\_TIM\_OC\_MspInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_OC_MspInit ( <i>TIM_HandleTypeDef</i> * htim)</b>          |
| Function Description | Initializes the TIM Output Compare MSP.                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li></ul> |

|               |                                                       |
|---------------|-------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>None.</li></ul> |
| Notes         | <ul style="list-style-type: none"><li>None.</li></ul> |

#### 41.2.26 HAL\_TIM\_OC\_MspDeInit

|                      |                                                                          |
|----------------------|--------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_OC_MspDeInit ( <i>TIM_HandleTypeDef</i> * htim)</b>      |
| Function Description | Deinitializes TIM Output Compare MSP.                                    |
| Parameters           | <ul style="list-style-type: none"><li><b>htim</b> : TIM handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                    |

#### 41.2.27 HAL\_TIM\_OC\_Start

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OC_Start ( <i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b>                                                                                                                                                                                                                                                                                                                                                                                               |
| Function Description | Starts the TIM Output Compare signal generation.                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"><li><b>htim</b> : : TIM Output Compare handle</li><li><b>Channel</b> : : TIM Channel to be enabled This parameter can be one of the following values:<ul style="list-style-type: none"><li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li><li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li><li>- <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li><li>- <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li></ul></li></ul> |
| Return values        | <ul style="list-style-type: none"><li><b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"><li>None.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                        |

#### 41.2.28 HAL\_TIM\_OC\_Stop

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OC_Stop ( <i>TIM_HandleTypeDef</i> * <i>htim</i>, <i>uint32_t</i> <i>Channel</i>)</b>                                                                                                                                                                                                                                                                                                                                               |
| Function Description | Stops the TIM Output Compare signal generation.                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> <li>• <b>Channel</b> : : TIM Channel to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <i>TIM_CHANNEL_1</i> TIM Channel 1 selected</li> <li>- <i>TIM_CHANNEL_2</i> TIM Channel 2 selected</li> <li>- <i>TIM_CHANNEL_3</i> TIM Channel 3 selected</li> <li>- <i>TIM_CHANNEL_4</i> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                        |

#### 41.2.29 HAL\_TIM\_OC\_Start\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OC_Start_IT ( <i>TIM_HandleTypeDef</i> * <i>htim</i>, <i>uint32_t</i> <i>Channel</i>)</b>                                                                                                                                                                                                                                                                                                                                             |
| Function Description | Starts the TIM Output Compare signal generation in interrupt mode.                                                                                                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM OC handle</li> <li>• <b>Channel</b> : : TIM Channel to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <i>TIM_CHANNEL_1</i> TIM Channel 1 selected</li> <li>- <i>TIM_CHANNEL_2</i> TIM Channel 2 selected</li> <li>- <i>TIM_CHANNEL_3</i> TIM Channel 3 selected</li> <li>- <i>TIM_CHANNEL_4</i> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                          |

#### 41.2.30 HAL\_TIM\_OC\_Stop\_IT

|                      |                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OC_Stop_IT ( <i>TIM_HandleTypeDef</i> * <i>htim</i>, <i>uint32_t</i> <i>Channel</i>)</b> |
| Function Description | Stops the TIM Output Compare signal generation in interrupt mode.                                                     |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Output Compare handle</li> <li>• <b>Channel</b> : : TIM Channel to be disabled This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> <li>– <b>TIM_CHANNEL_3</b> TIM Channel 3 selected</li> <li>– <b>TIM_CHANNEL_4</b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Notes         | • None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

#### 41.2.31 HAL\_TIM\_OC\_Start\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_OC_Start_DMA (</code><br><code>  <b>TIM_HandleTypeDef</b> * htim, uint32_t Channel, uint32_t *</code><br><code>  pData, uint16_t Length)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Function Description | Starts the TIM Output Compare signal generation in DMA mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Output Compare handle</li> <li>• <b>Channel</b> : : TIM Channel to be enabled This parameter can be one of the following values:             <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> <li>– <b>TIM_CHANNEL_3</b> TIM Channel 3 selected</li> <li>– <b>TIM_CHANNEL_4</b> TIM Channel 4 selected</li> </ul> </li> <li>• <b>pData</b> : The source Buffer address.</li> <li>• <b>Length</b> : The length of data to be transferred from memory to TIM peripheral</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | • None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

#### 41.2.32 HAL\_TIM\_OC\_Stop\_DMA

|                      |                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_OC_Stop_DMA (</code><br><code>  <b>TIM_HandleTypeDef</b> * htim, uint32_t Channel)</code>                                                 |
| Function Description | Stops the TIM Output Compare signal generation in DMA mode.                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Output Compare handle</li> <li>• <b>Channel</b> : : TIM Channel to be disabled This parameter can</li> </ul> |

|               |                                                      |
|---------------|------------------------------------------------------|
|               | be one of the following values:                      |
|               | – <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected |
|               | – <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected |
|               | – <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected |
|               | – <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected |
| Return values | • <b>HAL status</b>                                  |
| Notes         | • None.                                              |

#### 41.2.33 HAL\_TIM\_PWM\_Init

|                      |                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_PWM_Init (</b><br><b><i>TIM_HandleTypeDef</i> * htim)</b>                                                    |
| Function Description | Initializes the TIM PWM Time Base according to the specified parameters in the <i>TIM_HandleTypeDef</i> and create the associated handle. |
| Parameters           | • <b>htim</b> : TIM handle                                                                                                                |
| Return values        | • <b>HAL status</b>                                                                                                                       |
| Notes                | • None.                                                                                                                                   |

#### 41.2.34 HAL\_TIM\_PWM\_DeInit

|                      |                                                                                          |
|----------------------|------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_PWM_DeInit (</b><br><b><i>TIM_HandleTypeDef</i> * htim)</b> |
| Function Description | Deinitializes the TIM peripheral.                                                        |
| Parameters           | • <b>htim</b> : TIM handle                                                               |
| Return values        | • <b>HAL status</b>                                                                      |
| Notes                | • None.                                                                                  |

#### 41.2.35 HAL\_TIM\_PWM\_MspInit

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_PWM_MspInit ( <i>TIM_HandleTypeDef</i> * htim)</b>           |
| Function Description | Initializes the TIM PWM MSP.                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |

#### 41.2.36 HAL\_TIM\_PWM\_MspDeInit

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_PWM_MspDeInit ( <i>TIM_HandleTypeDef</i> * htim)</b>         |
| Function Description | Deinitializes TIM PWM MSP.                                                   |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |

#### 41.2.37 HAL\_TIM\_PWM\_Start

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_PWM_Start ( <i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b>                                                                                                                                                                                                                                                                                                                                                                  |
| Function Description | Starts the PWM signal generation.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> <li>– <b>TIM_CHANNEL_3</b> TIM Channel 3 selected</li> <li>– <b>TIM_CHANNEL_4</b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                        |

### 41.2.38 HAL\_TIM\_PWM\_Stop

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_PWM_Stop (</b><br><i><b>TIM_HandleTypeDef</b></i> * htim, uint32_t Channel)                                                                                                                                                                                                                                                                                                                                                          |
| Function Description | Stops the PWM signal generation.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> <li>• <b>Channel</b> : : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>- <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> <li>- <b>TIM_CHANNEL_3</b> TIM Channel 3 selected</li> <li>- <b>TIM_CHANNEL_4</b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                         |

### 41.2.39 HAL\_TIM\_PWM\_Start\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_PWM_Start_IT (</b><br><i><b>TIM_HandleTypeDef</b></i> * htim, uint32_t Channel)                                                                                                                                                                                                                                                                                                                                                     |
| Function Description | Starts the PWM signal generation in interrupt mode.                                                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> <li>• <b>Channel</b> : : TIM Channel to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>- <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> <li>- <b>TIM_CHANNEL_3</b> TIM Channel 3 selected</li> <li>- <b>TIM_CHANNEL_4</b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                        |

### 41.2.40 HAL\_TIM\_PWM\_Stop\_IT

|               |                                                |
|---------------|------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_TIM_PWM_Stop_IT (</b> |
|---------------|------------------------------------------------|

***TIM\_HandleTypeDef \* htim, uint32\_t Channel)***

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Stops the PWM signal generation in interrupt mode.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> <li>• <b>Channel</b> : : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> <li>- <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li> <li>- <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                     |

#### 41.2.41 HAL\_TIM\_PWM\_Start\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_PWM_Start_DMA (</b><br><b><i>TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData, uint16_t Length)</i></b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Function Description | Starts the TIM PWM signal generation in DMA mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> <li>- <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li> <li>- <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li> </ul> </li> <li>• <b>pData</b> : The source Buffer address.</li> <li>• <b>Length</b> : The length of data to be transferred from memory to TIM peripheral</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

#### 41.2.42 HAL\_TIM\_PWM\_Stop\_DMA

|               |                                                                                                              |
|---------------|--------------------------------------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_TIM_PWM_Stop_DMA (</b><br><b><i>TIM_HandleTypeDef * htim, uint32_t Channel)</i></b> |
|---------------|--------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Stops the TIM PWM signal generation in DMA mode.                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> <li>• <b>Channel</b> : : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> <li>– <b>TIM_CHANNEL_3</b> TIM Channel 3 selected</li> <li>– <b>TIM_CHANNEL_4</b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                         |

#### 41.2.43 HAL\_TIM\_IC\_Init

|                      |                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_IC_Init ( <i>TIM_HandleTypeDef</i> *<br/>htim)</b>                                                                     |
| Function Description | Initializes the TIM Input Capture Time base according to the specified parameters in the <i>TIM_HandleTypeDef</i> and create the associated handle. |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM Input Capture handle</li> </ul>                                                          |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                           |

#### 41.2.44 HAL\_TIM\_IC\_DeInit

|                      |                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_IC_DeInit ( <i>TIM_HandleTypeDef</i><br/>* htim)</b>          |
| Function Description | Deinitializes the TIM peripheral.                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM Input Capture handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                  |

#### 41.2.45 HAL\_TIM\_IC\_MspInit

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_IC_MspInit ( <i>TIM_HandleTypeDef</i> * htim)</b>            |
| Function Description | Initializes the TIM INput Capture MSP.                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |

#### 41.2.46 HAL\_TIM\_IC\_MspDeInit

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_IC_MspDeInit ( <i>TIM_HandleTypeDef</i> * htim)</b>          |
| Function Description | Deinitializes TIM Input Capture MSP.                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |

#### 41.2.47 HAL\_TIM\_IC\_Start

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_IC_Start ( <i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b>                                                                                                                                                                                                                                                                                                                                                                                 |
| Function Description | Starts the TIM Input Capture measurement.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Input Capture handle</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>- <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> <li>- <b>TIM_CHANNEL_3</b> TIM Channel 3 selected</li> <li>- <b>TIM_CHANNEL_4</b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                      |

#### 41.2.48 HAL\_TIM\_IC\_Stop

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_IC_Stop ( <i>TIM_HandleTypeDef</i> * <i>htim</i>, uint32_t <i>Channel</i>)</b>                                                                                                                                                                                                                                                                                                                                                                         |
| Function Description | Stops the TIM Input Capture measurement.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li><li>• <b>Channel</b> : : TIM Channels to be disabled This parameter can be one of the following values:<ul style="list-style-type: none"><li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li><li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li><li>- <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li><li>- <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li></ul></li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                             |

#### 41.2.49 HAL\_TIM\_IC\_Start\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_IC_Start_IT ( <i>TIM_HandleTypeDef</i> * <i>htim</i>, uint32_t <i>Channel</i>)</b>                                                                                                                                                                                                                                                                                                                                                                                  |
| Function Description | Starts the TIM Input Capture measurement in interrupt mode.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM Input Capture handle</li><li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values:<ul style="list-style-type: none"><li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li><li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li><li>- <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li><li>- <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li></ul></li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                          |

#### 41.2.50 HAL\_TIM\_IC\_Stop\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_IC_Stop_IT (</code><br><code>TIM_HandleTypeDef * htim, uint32_t Channel)</code>                                                                                                                                                                                                                                                                                                                                                                           |
| Function Description | Stops the TIM Input Capture measurement in interrupt mode.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> <li>• <b>Channel</b> : : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <code>TIM_CHANNEL_1</code> TIM Channel 1 selected</li> <li>- <code>TIM_CHANNEL_2</code> TIM Channel 2 selected</li> <li>- <code>TIM_CHANNEL_3</code> TIM Channel 3 selected</li> <li>- <code>TIM_CHANNEL_4</code> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                 |

#### 41.2.51 HAL\_TIM\_IC\_Start\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_IC_Start_DMA (</code><br><code>TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t *</code><br><code>pData, uint16_t Length)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Function Description | Starts the TIM Input Capture measurement on in DMA mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Input Capture handle</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <code>TIM_CHANNEL_1</code> TIM Channel 1 selected</li> <li>- <code>TIM_CHANNEL_2</code> TIM Channel 2 selected</li> <li>- <code>TIM_CHANNEL_3</code> TIM Channel 3 selected</li> <li>- <code>TIM_CHANNEL_4</code> TIM Channel 4 selected</li> </ul> </li> <li>• <b>pData</b> : The destination Buffer address.</li> <li>• <b>Length</b> : The length of data to be transferred from TIM peripheral to memory.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

#### 41.2.52 HAL\_TIM\_IC\_Stop\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_IC_Stop_DMA (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b>                                                                                                                                                                                                                                                                                                                                                                                                 |
| Function Description | Stops the TIM Input Capture measurement on in DMA mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Input Capture handle</li> <li>• <b>Channel</b> : : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>– <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> <li>– <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li> <li>– <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

#### 41.2.53 HAL\_TIM\_OnePulse\_Init

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OnePulse_Init (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t OnePulseMode)</b>                                                                                                                                                                                                                                                                                             |
| Function Description | Initializes the TIM One Pulse Time Base according to the specified parameters in the <i>TIM_HandleTypeDef</i> and create the associated handle.                                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM OnePulse handle</li> <li>• <b>OnePulseMode</b> : Select the One pulse mode. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b><i>TIM_OPmode_SINGLE</i></b> Only one pulse will be generated.</li> <li>– <b><i>TIM_OPmode_Repetitive</i></b> Repetitive pulses wil be generated.</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                      |

#### 41.2.54 HAL\_TIM\_OnePulse\_DeInit

|                      |                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OnePulse_DeInit (</b><br><b><i>TIM_HandleTypeDef</i> * htim)</b> |
| Function Description | Deinitializes the TIM One Pulse.                                                              |

|               |                                                                                      |
|---------------|--------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM One Pulse handle</li></ul> |
| Return values | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                  |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>                              |

#### 41.2.55 HAL\_TIM\_OnePulse\_MspInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_OnePulse_MspInit ( <i>TIM_HandleTypeDef</i> * htim)</b>    |
| Function Description | Initializes the TIM One Pulse MSP.                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 41.2.56 HAL\_TIM\_OnePulse\_MspDeInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_OnePulse_MspDeInit ( <i>TIM_HandleTypeDef</i> * htim)</b>  |
| Function Description | Deinitializes TIM One Pulse MSP.                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 41.2.57 HAL\_TIM\_OnePulse\_Start

|                      |                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OnePulse_Start ( <i>TIM_HandleTypeDef</i> * htim, uint32_t OutputChannel)</b> |
| Function Description | Starts the TIM One Pulse signal generation.                                                                |

|               |                                                                                                                                                                                                                                                                                                                                                                          |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li><b>htim</b> : : TIM One Pulse handle</li> <li><b>OutputChannel</b> : : TIM Channels to be enabled This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                      |
| Notes         | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                                                  |

#### 41.2.58 HAL\_TIM\_OnePulse\_Stop

|                      |                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OnePulse_Stop (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t OutputChannel)</b>                                                                                                                                                                                                                                                      |
| Function Description | Stops the TIM One Pulse signal generation.                                                                                                                                                                                                                                                                                                                               |
| Parameters           | <ul style="list-style-type: none"> <li><b>htim</b> : : TIM One Pulse handle</li> <li><b>OutputChannel</b> : : TIM Channels to be disable This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                                                  |

#### 41.2.59 HAL\_TIM\_OnePulse\_Start\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OnePulse_Start_IT (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t OutputChannel)</b>                                                                                                                                                                                                                                                  |
| Function Description | Starts the TIM One Pulse signal generation in interrupt mode.                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li><b>htim</b> : : TIM One Pulse handle</li> <li><b>OutputChannel</b> : : TIM Channels to be enabled This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                                                                  |

#### 41.2.60 HAL\_TIM\_OnePulse\_Stop\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OnePulse_Stop_IT (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t OutputChannel)</b>                                                                                                                                                                                                                                                           |
| Function Description | Stops the TIM One Pulse signal generation in interrupt mode.                                                                                                                                                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM One Pulse handle</li> <li>• <b>OutputChannel</b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                        |

#### 41.2.61 HAL\_TIM\_Encoder\_Init

|                      |                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Encoder_Init (</b><br><b><i>TIM_HandleTypeDef</i> * htim, <i>TIM_Encoder_InitTypeDef</i> * sConfig)</b>                                     |
| Function Description | Initializes the TIM Encoder Interface and create the associated handle.                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM Encoder Interface handle</li> <li>• <b>sConfig</b> : TIM Encoder Interface configuration structure</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                |

#### 41.2.62 HAL\_TIM\_Encoder\_DelInit

|               |                                                                                               |
|---------------|-----------------------------------------------------------------------------------------------|
| Function Name | <b>HAL_StatusTypeDef HAL_TIM_Encoder_DelInit (</b><br><b><i>TIM_HandleTypeDef</i> * htim)</b> |
|---------------|-----------------------------------------------------------------------------------------------|

---

|                      |                                                                                    |
|----------------------|------------------------------------------------------------------------------------|
| Function Description | Deinitializes the TIM Encoder interface.                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM Encoder handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                            |

#### 41.2.63 HAL\_TIM\_Encoder\_MspInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_Encoder_MspInit ( <i>TIM_HandleTypeDef</i> * htim)</b>     |
| Function Description | Initializes the TIM Encoder Interface MSP.                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 41.2.64 HAL\_TIM\_Encoder\_MspDeInit

|                      |                                                                            |
|----------------------|----------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_Encoder_MspDeInit ( <i>TIM_HandleTypeDef</i> * htim)</b>   |
| Function Description | Deinitializes TIM Encoder Interface MSP.                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                    |

#### 41.2.65 HAL\_TIM\_Encoder\_Start

|                      |                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Encoder_Start ( <i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b> |
| Function Description | Starts the TIM Encoder Interface.                                                                   |

---

|               |                                                                                                                                                                                                                                                                                                                                                                                |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Encoder Interface handle</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                          |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                      |

#### 41.2.66 HAL\_TIM\_Encoder\_Stop

|                      |                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Encoder_Stop (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b>                                                                                                                                                                                                                                                                    |
| Function Description | Stops the TIM Encoder Interface.                                                                                                                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Encoder Interface handle</li> <li>• <b>Channel</b> : : TIM Channels to be disabled This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                       |

#### 41.2.67 HAL\_TIM\_Encoder\_Start\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Encoder_Start_IT (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b>                                                                                                                                                                                                                                                               |
| Function Description | Starts the TIM Encoder Interface in interrupt mode.                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Encoder Interface handle</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values:           <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                      |

#### 41.2.68 HAL\_TIM\_Encoder\_Stop\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Encoder_Stop_IT (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b>                                                                                                                                                                                                                                                                     |
| Function Description | Stops the TIM Encoder Interface in interrupt mode.                                                                                                                                                                                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Encoder Interface handle</li> <li>• <b>Channel</b> : : TIM Channels to be disabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>– <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                           |

#### 41.2.69 HAL\_TIM\_Encoder\_Start\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Encoder_Start_DMA (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t Channel, uint32_t * pData1, uint32_t * pData2, uint16_t Length)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Function Description | Starts the TIM Encoder Interface in DMA mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Encoder Interface handle</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>– <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> </ul> </li> <li>• <b>pData1</b> : The destination Buffer address for IC1.</li> <li>• <b>pData2</b> : The destination Buffer address for IC2.</li> <li>• <b>Length</b> : The length of data to be transferred from TIM peripheral to memory.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

#### 41.2.70 HAL\_TIM\_Encoder\_Stop\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_Encoder_Stop_DMA (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t Channel)</b>                                                                                                                                                                                                                                                                   |
| Function Description | Stops the TIM Encoder Interface in DMA mode.                                                                                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM Encoder Interface handle</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                          |

#### 41.2.71 HAL\_TIM\_IRQHandler

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_IRQHandler (</b><br><b><i>TIM_HandleTypeDef</i> * htim)</b>  |
| Function Description | This function handles TIM interrupts requests.                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                    |

#### 41.2.72 HAL\_TIM\_OC\_ConfigChannel

|                      |                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_OC_ConfigChannel (</b><br><b><i>TIM_HandleTypeDef</i> * htim, <i>TIM_OC_InitTypeDef</i> * sConfig,</b><br><b>uint32_t Channel)</b>                                                                                                                |
| Function Description | Initializes the TIM Output Compare Channels according to the specified parameters in the <i>TIM_OC_InitTypeDef</i> .                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM Output Compare handle</li> <li>• <b>sConfig</b> : TIM Output Compare configuration structure</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values:</li> </ul> |

---

|               |                                                                                                                                                                                                                                                                                                  |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> <li>- <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li> <li>- <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                            |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                        |

#### 41.2.73 HAL\_TIM\_IC\_ConfigChannel

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_IC_ConfigChannel (</b><br><b><i>TIM_HandleTypeDef</i> * htim, <i>TIM_IC_InitTypeDef</i> * sConfig,</b><br><b>uint32_t Channel)</b>                                                                                                                                                                                                                                                                                                                                                                                                     |
| Function Description | Initializes the TIM Input Capture Channels according to the specified parameters in the <i>TIM_IC_InitTypeDef</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM IC handle</li> <li>• <b>sConfig</b> : TIM Input Capture configuration structure</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> <li>- <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li> <li>- <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

#### 41.2.74 HAL\_TIM\_PWM\_ConfigChannel

|                      |                                                                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_PWM_ConfigChannel (</b><br><b><i>TIM_HandleTypeDef</i> * htim, <i>TIM_OC_InitTypeDef</i> * sConfig,</b><br><b>uint32_t Channel)</b>                                                                                     |
| Function Description | Initializes the TIM PWM channels according to the specified parameters in the <i>TIM_OC_InitTypeDef</i> .                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> <li>• <b>sConfig</b> : TIM PWM configuration structure</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values:</li> </ul> |

---

|               |                                                                                                                                                                                                                                                                                                  |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> <li>- <b><i>TIM_CHANNEL_3</i></b> TIM Channel 3 selected</li> <li>- <b><i>TIM_CHANNEL_4</i></b> TIM Channel 4 selected</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                            |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                        |

#### 41.2.75 HAL\_TIM\_OnePulse\_ConfigChannel

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_OnePulse_ConfigChannel (</code><br><b><i>TIM_HandleTypeDef</i></b> * <i>htim</i> , <b><i>TIM_OnePulse_InitTypeDef</i></b> *<br><i>sConfig</i> , <i>uint32_t</i> <i>OutputChannel</i> , <i>uint32_t</i> <i>InputChannel</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Function Description | Initializes the TIM One Pulse Channels according to the specified parameters in the <b><i>TIM_OnePulse_InitTypeDef</i></b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b><i>htim</i></b> : TIM One Pulse handle</li> <li>• <b><i>sConfig</i></b> : TIM One Pulse configuration structure</li> <li>• <b><i>OutputChannel</i></b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> </ul> </li> <li>• <b><i>InputChannel</i></b> : : TIM Channels to be enabled This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <b><i>TIM_CHANNEL_1</i></b> TIM Channel 1 selected</li> <li>- <b><i>TIM_CHANNEL_2</i></b> TIM Channel 2 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

#### 41.2.76 HAL\_TIM\_DMABurst\_WriteStart

|                      |                                                                                                                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_DMABurst_WriteStart (</code><br><b><i>TIM_HandleTypeDef</i></b> * <i>htim</i> , <i>uint32_t</i> <i>BurstBaseAddress</i> ,<br><i>uint32_t</i> <i>BurstRequestSrc</i> , <i>uint32_t</i> * <i>BurstBuffer</i> , <i>uint32_t</i><br><i>BurstLength</i> ) |
| Function Description | Configure the DMA Burst to transfer Data from the memory to the TIM peripheral.                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b><i>htim</i></b> : TIM handle</li> </ul>                                                                                                                                                                                                  |

- **BurstBaseAddress** : TIM Base address from when the DMA will starts the Data write This parameters can be on of the following values:
  - ***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 parameters can be on 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**

Notes

- None.

#### 41.2.77 HAL\_TIM\_DMABurst\_WriteStop

|                      |                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_DMABurst_WriteStop (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t BurstRequestSrc)</b>                          |
| Function Description | Stops the TIM DMA Burst mode.                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> <li>• <b>BurstRequestSrc</b> : TIM DMA Request sources to disable</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                 |

## Notes

- None.

### 41.2.78 HAL\_TIM\_DMABurst\_ReadStart

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_DMABurst_ReadStart (<br/>    <b>TIM_HandleTypeDef</b> * htim, uint32_t BurstBaseAddress,<br/>    uint32_t BurstRequestSrc, uint32_t * BurstBuffer, uint32_t<br/>    BurstLength)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Function Description | Configure the DMA Burst to transfer Data from the TIM peripheral to the memory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> <li>• <b>BurstBaseAddress</b> : TIM Base address from when the DMA will starts the Data read This parameters can be one of the following values: <ul style="list-style-type: none"> <li>- <b>TIM_DMABase_CR1</b></li> <li>- <b>TIM_DMABase_CR2</b></li> <li>- <b>TIM_DMABase_SMCR</b></li> <li>- <b>TIM_DMABase_DIER</b></li> <li>- <b>TIM_DMABase_SR</b></li> <li>- <b>TIM_DMABase_EGR</b></li> <li>- <b>TIM_DMABase_CCMR1</b></li> <li>- <b>TIM_DMABase_CCMR2</b></li> <li>- <b>TIM_DMABase_CCER</b></li> <li>- <b>TIM_DMABase_CNT</b></li> <li>- <b>TIM_DMABase_PSC</b></li> <li>- <b>TIM_DMABase_ARR</b></li> <li>- <b>TIM_DMABase_CCR1</b></li> <li>- <b>TIM_DMABase_CCR2</b></li> <li>- <b>TIM_DMABase_CCR3</b></li> <li>- <b>TIM_DMABase_CCR4</b></li> <li>- <b>TIM_DMABase_DCR</b></li> </ul> </li> <li>• <b>BurstRequestSrc</b> : TIM DMA Request sources This parameters can be one of the following values: <ul style="list-style-type: none"> <li>- <b>TIM_DMA_UPDATE</b> TIM update Interrupt source</li> <li>- <b>TIM_DMA_CC1</b> TIM Capture Compare 1 DMA source</li> <li>- <b>TIM_DMA_CC2</b> TIM Capture Compare 2 DMA source</li> <li>- <b>TIM_DMA_CC3</b> TIM Capture Compare 3 DMA source</li> <li>- <b>TIM_DMA_CC4</b> TIM Capture Compare 4 DMA source</li> <li>- <b>TIM_DMA_TRIGGER</b> TIM Trigger DMA source</li> </ul> </li> <li>• <b>BurstBuffer</b> : The Buffer address.</li> <li>• <b>BurstLength</b> : DMA Burst length. This parameter can be one value between: <b>TIM_DMABurstLength_1Transfer</b> and <b>TIM_DMABurstLength_18Transfers</b>.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

## Notes

- None.

#### 41.2.79 HAL\_TIM\_DMABurst\_ReadStop

|                      |                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_DMABurst_ReadStop (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t BurstRequestSrc)</b>                         |
| Function Description | Stop the DMA burst reading.                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li><li>• <b>BurstRequestSrc</b> : TIM DMA Request sources to disable.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                               |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                           |

#### 41.2.80 HAL\_TIM\_GenerateEvent

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIM_GenerateEvent (</b><br><b><i>TIM_HandleTypeDef</i> * htim, uint32_t EventSource)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Function Description | Generate a software event.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li><li>• <b>EventSource</b> : specifies the event source. This parameter can be one of the following values:<ul style="list-style-type: none"><li>– <b><i>TIM_EventSource_Update</i></b> Timer update Event source</li><li>– <b><i>TIM_EventSource_CC1</i></b> Timer Capture Compare 1 Event source</li><li>– <b><i>TIM_EventSource_CC2</i></b> Timer Capture Compare 2 Event source</li><li>– <b><i>TIM_EventSource_CC3</i></b> Timer Capture Compare 3 Event source</li><li>– <b><i>TIM_EventSource_CC4</i></b> Timer Capture Compare 4 Event source</li><li>– <b><i>TIM_EventSource_Trigger</i></b> Timer Trigger Event source</li></ul></li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Notes                | <ul style="list-style-type: none"><li>• TBC can only generate an update event.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

#### 41.2.81 HAL\_TIM\_ConfigOCrefClear

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_ConfigOCrefClear (</code><br><code> <b>TIM_HandleTypeDef</b> * htim, <b>TIM_ClearInputConfigTypeDef</b> *</code><br><code>    sClearInputConfig, uint32_t Channel)</code>                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Function Description | Configures the OCRef clear feature.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> <li>• <b>sClearInputConfig</b> : pointer to a <code>TIM_ClearInputConfigTypeDef</code> structure that contains the OCREF clear feature and parameters for the TIM peripheral.</li> <li>• <b>Channel</b> : specifies the TIM Channel This parameter can be one of the following values: <ul style="list-style-type: none"> <li>- <code>TIM_CHANNEL_1</code> TIM Channel 1</li> <li>- <code>TIM_CHANNEL_2</code> TIM Channel 2</li> <li>- <code>TIM_CHANNEL_3</code> TIM Channel 3</li> <li>- <code>TIM_CHANNEL_4</code> TIM Channel 4</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

#### 41.2.82 HAL\_TIM\_ConfigClockSource

|                      |                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_ConfigClockSource (</code><br><code> <b>TIM_HandleTypeDef</b> * htim, <b>TIM_ClockConfigTypeDef</b> *</code><br><code>    sClockSourceConfig)</code>                                                              |
| Function Description | Configures the clock source to be used.                                                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle</li> <li>• <b>sClockSourceConfig</b> : pointer to a <code>TIM_ClockConfigTypeDef</code> structure that contains the clock source information for the TIM peripheral.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                             |

Notes

#### 41.2.83 HAL\_TIM\_ConfigTI1Input

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_ConfigTI1Input (<br/>TIM_HandleTypeDef * htim, uint32_t TI1_Selection)</code>                                                                                                                                                                                                                                                                                                                                                                                                |
| 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"> <li>• <b>htim</b> : TIM handle.</li> <li>• <b>TI1_Selection</b> : 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"> <li>– <b>TIM_TI1SELECTION_CH1</b> The TIMx_CH1 pin is connected to TI1 input</li> <li>– <b>TIM_TI1SELECTION_XORCOMBINATION</b> The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

#### 41.2.84 HAL\_TIM\_SlaveConfigSynchronization

|                      |                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_TIM_SlaveConfigSynchronization (<br/>TIM_HandleTypeDef * htim, TIM_SlaveConfigTypeDef *<br/>sSlaveConfig)</code>                                                                                                                                                                                                                   |
| Function Description | Configures the TIM in Slave mode.                                                                                                                                                                                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle.</li> <li>• <b>sSlaveConfig</b> : 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).</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                      |

#### 41.2.85 HAL\_TIM\_ReadCapturedValue

|               |                                                                                                   |
|---------------|---------------------------------------------------------------------------------------------------|
| Function Name | <code>uint32_t HAL_TIM_ReadCapturedValue ( TIM_HandleTypeDef *<br/>htim, uint32_t Channel)</code> |
|---------------|---------------------------------------------------------------------------------------------------|

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Read the captured value from Capture Compare unit.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle.</li> <li>• <b>Channel</b> : : TIM Channels to be enabled This parameter can be one of the following values:               <ul style="list-style-type: none"> <li>– <b>TIM_CHANNEL_1</b> TIM Channel 1 selected</li> <li>– <b>TIM_CHANNEL_2</b> TIM Channel 2 selected</li> <li>– <b>TIM_CHANNEL_3</b> TIM Channel 3 selected</li> <li>– <b>TIM_CHANNEL_4</b> TIM Channel 4 selected</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>Captured value</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                     |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                     |

#### 41.2.86 HAL\_TIM\_PeriodElapsedCallback

|                      |                                                                                |
|----------------------|--------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_PeriodElapsedCallback ( <i>TIM_HandleTypeDef</i> * htim)</b>   |
| Function Description | Period elapsed callback in non blocking mode.                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                      |

#### 41.2.87 HAL\_TIM\_OC\_DelayElapsedCallback

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_OC_DelayElapsedCallback ( <i>TIM_HandleTypeDef</i> * htim)</b>    |
| Function Description | Output Compare callback in non blocking mode.                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : : TIM OC handle</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                         |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                         |

#### 41.2.88 HAL\_TIM\_IC\_CaptureCallback

|                      |                                                                                  |
|----------------------|----------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_IC_CaptureCallback ( <i>TIM_HandleTypeDef</i> * <i>htim</i>)</b> |
| Function Description | Input Capture callback in non blocking mode.                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM IC handle</li></ul>  |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                          |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                          |

#### 41.2.89 HAL\_TIM\_PWM\_PulseFinishedCallback

|                      |                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_PWM_PulseFinishedCallback ( <i>TIM_HandleTypeDef</i> * <i>htim</i>)</b> |
| Function Description | PWM Pulse finished callback in non blocking mode.                                       |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li></ul>            |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                 |

#### 41.2.90 HAL\_TIM\_TriggerCallback

|                      |                                                                               |
|----------------------|-------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_TriggerCallback ( <i>TIM_HandleTypeDef</i> * <i>htim</i>)</b> |
| Function Description | Hall Trigger detection callback in non blocking mode.                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li></ul>  |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                       |

#### 41.2.91 HAL\_TIM\_ErrorCallback

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_TIM_ErrorCallback ( <i>TIM_HandleTypeDef</i> * htim)</b>         |
| Function Description | Timer error callback in non blocking mode.                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : : TIM handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                      |

#### 41.2.92 HAL\_TIM\_Base\_GetState

|                      |                                                                                      |
|----------------------|--------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_TIM_StateTypeDef HAL_TIM_Base_GetState ( <i>TIM_HandleTypeDef</i> * htim)</b> |
| Function Description | Return the TIM Base state.                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM Base handle</li></ul>      |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                              |

#### 41.2.93 HAL\_TIM\_OC\_GetState

|                      |                                                                                          |
|----------------------|------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_TIM_StateTypeDef HAL_TIM_OC_GetState ( <i>TIM_HandleTypeDef</i> * htim)</b>       |
| Function Description | Return the TIM OC state.                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM Ouput Compare handle</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                  |

#### 41.2.94 HAL\_TIM\_PWM\_GetState

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_TIM_StateTypeDef HAL_TIM_PWM_GetState (</b><br><b><i>TIM_HandleTypeDef * htim</i></b> ) |
| Function Description | Return the TIM PWM state.                                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM handle</li></ul>                     |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                             |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                        |

#### 41.2.95 HAL\_TIM\_IC\_GetState

|                      |                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_TIM_StateTypeDef HAL_TIM_IC_GetState (</b><br><b><i>TIM_HandleTypeDef * htim</i></b> ) |
| Function Description | Return the TIM Input Capture state.                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM IC handle</li></ul>                 |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                            |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                       |

#### 41.2.96 HAL\_TIM\_OnePulse\_GetState

|                      |                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_TIM_StateTypeDef HAL_TIM_OnePulse_GetState (</b><br><b><i>TIM_HandleTypeDef * htim</i></b> ) |
| Function Description | Return the TIM One Pulse Mode state.                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>htim</b> : TIM OPM handle</li></ul>                      |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                             |

## 41.2.97 HAL\_TIM\_Encoder\_GetState

|                      |                                                                                                                |
|----------------------|----------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_TIM_StateTypeDef HAL_TIM_Encoder_GetState (</code><br><i><code>TIM_HandleTypeDef * htim</code></i> ) |
| Function Description | Return the TIM Encoder Mode state.                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <code>htim</code> : TIM Encoder handle</li> </ul>                     |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul>                                           |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                      |

## 41.3 TIM Firmware driver defines

### 41.3.1 TIM

TIM

***TIM AOE Bit Set Reset***

- `TIM_AUTOMATICOUTPUT_ENABLE`
- `TIM_AUTOMATICOUTPUT_DISABLE`
- `IS_TIM_AUTOMATIC_OUTPUT_STATE`

***TIM Channel***

- `TIM_CHANNEL_1`
- `TIM_CHANNEL_2`
- `TIM_CHANNEL_3`
- `TIM_CHANNEL_4`
- `TIM_CHANNEL_ALL`
- `IS_TIM_CHANNELS`
- `IS_TIM_PWM_CHANNELS`
- `IS_TIM_OPM_CHANNELS`

***TIM ClearInput Filter***

- `IS_TIM_CLEARINPUT_FILTER`

***TIM ClearInput Polarity***

- `TIM_CLEARINPUTPOLARITY_INVERTED`  
Polarity for ETRx pin
- `TIM_CLEARINPUTPOLARITY_NONINVERTED`  
Polarity for ETRx pin
- `IS_TIM_CLEARINPUT_POLARITY`

***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.
- **IS\_TIM\_CLEARINPUT\_PRESCALER**

#### ***TIM ClearInput Source***

- **TIM\_CLEARINPUTSOURCE\_ETR**
- **TIM\_CLEARINPUTSOURCE\_OCREFCLR**
- **TIM\_CLEARINPUTSOURCE\_NONE**
- **IS\_TIM\_CLEARINPUT\_SOURCE**

#### ***TIM ClockDivision***

- **TIM\_CLOCKDIVISION\_DIV1**
- **TIM\_CLOCKDIVISION\_DIV2**
- **TIM\_CLOCKDIVISION\_DIV4**
- **IS\_TIM\_CLOCKDIVISION\_DIV**

#### ***TIM Clock Filter***

- **IS\_TIM\_CLOCKFILTER**

#### ***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
- **IS\_TIM\_CLOCKPOLARITY**

#### ***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.
- **IS\_TIM\_CLOCKPRESCALER**

#### ***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`
- `IS_TIM_CLOCKSOURCE`

#### ***TIM Counter Mode***

- `TIM_COUNTERMODE_UP`
- `TIM_COUNTERMODE_DOWN`
- `TIM_COUNTERMODE_CENTERALIGNED1`
- `TIM_COUNTERMODE_CENTERALIGNED2`
- `TIM_COUNTERMODE_CENTERALIGNED3`
- `IS_TIM_COUNTER_MODE`

#### ***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`
- `IS_TIM_DMA_BASE`

#### ***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`

- **IS\_TIM\_DMA\_LENGTH**

#### **TIM DMA sources**

- **TIM\_DMA\_UPDATE**
- **TIM\_DMA\_CC1**
- **TIM\_DMA\_CC2**
- **TIM\_DMA\_CC3**
- **TIM\_DMA\_CC4**
- **TIM\_DMA\_TRIGGER**
- **IS\_TIM\_DMA\_SOURCE**

#### **TIM Encoder Mode**

- **TIM\_ENCODERMODE\_TI1**
- **TIM\_ENCODERMODE\_TI2**
- **TIM\_ENCODERMODE\_TI12**
- **IS\_TIM\_ENCODER\_MODE**

#### **TIM ETR Polarity**

- **TIM\_ETRPOLARITY\_INVERTED**  
Polarity for ETR source
- **TIM\_ETRPOLARITY\_NONINVERTED**  
Polarity for ETR source

#### **TIM ETR Prescaler**

- **TIM\_ETRPRESCALER\_DIV1**  
No prescaler is used
- **TIM\_ETRPRESCALER\_DIV2**  
ETR input source is divided by 2
- **TIM\_ETRPRESCALER\_DIV4**  
ETR input source is divided by 4
- **TIM\_ETRPRESCALER\_DIV8**  
ETR input source is divided by 8

#### **TIM Event Source**

- **TIM\_EventSource\_Update**
- **TIM\_EventSource\_CC1**
- **TIM\_EventSource\_CC2**
- **TIM\_EventSource\_CC3**
- **TIM\_EventSource\_CC4**
- **TIM\_EventSource\_Trigger**
- **IS\_TIM\_EVENT\_SOURCE**

#### **TIM Exported Macros**

- **\_\_HAL\_TIM\_RESET\_HANDLE\_STATE**  
**Description:** 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:** Enable the specified TIM interrupt.  
**Parameters:** `__HANDLE__`: TIM handle `__INTERRUPT__`: specifies the TIM interrupt sources to be enabled or disabled.  
**Return value:**None
  - **`__HAL_TIM_ENABLE_DMA`**  
**Description:** Enable the specified DMA Channel.  
**Parameters:** `__HANDLE__`: TIM handle `__DMA__`: specifies the DMA Channel to be enabled or disabled.  
**Return value:**None
  - **`__HAL_TIM_DISABLE_IT`**  
**Description:** Disable the specified TIM interrupt.  
**Parameters:** `__HANDLE__`: TIM handle `__INTERRUPT__`: specifies the TIM interrupt sources to be enabled or disabled.  
**Return value:**None
  - **`__HAL_TIM_DISABLE_DMA`**  
**Description:** Disable the specified DMA Channel.  
**Parameters:** `__HANDLE__`: TIM handle `__DMA__`: specifies the DMA Channel to be enabled or disabled.  
**Return value:**None
  - **`__HAL_TIM_GET_FLAG`**  
**Description:** Get the TIM Channel pending flags.  
**Parameters:** `__HANDLE__`: TIM handle `__FLAG__`: Get the specified flag.  
**Return value:**The state of FLAG (SET or RESET).
  - **`__HAL_TIM_CLEAR_FLAG`**  
**Description:** Clear the TIM Channel pending flags.  
**Parameters:** `__HANDLE__`: TIM handle `__FLAG__`: specifies the flag to clear.  
**Return value:**None
  - **`__HAL_TIM_GET_ITSTATUS`**  
**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_DIRECTION_STATUS`**  
**Description:** TIM counter direction.  
**Parameters:** `__HANDLE__`: TIM handle
  - **`__HAL_TIM_PRESCALER`**  
**Description:** Set TIM prescaler.  
**Parameters:** `__HANDLE__`: TIM handle `__PRESC__`: specifies the prescaler value.  
**Return value:**None
  - **`__HAL_TIM_SetICPrescalerValue`**  
**Description:** Set TIM IC prescaler.  
**Parameters:** `__HANDLE__`: TIM handle `__CHANNEL__`: specifies TIM Channel `__ICPSC__`: specifies the prescaler value.  
**Return value:**None
  - **`__HAL_TIM_ResetICPrescalerValue`**  
**Description:** Reset TIM IC prescaler.

- Parameters:** `__HANDLE__`: TIM handle `__CHANNEL__`: specifies TIM Channel  
**Return value:**None
- **`__HAL_TIM_SetCompare`**  
**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_GetCompare`**  
**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_SetCounter`**  
**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_GetCounter`**  
**Description:** Gets the TIM Counter Register value on runtime.  
**Parameters:** `__HANDLE__`: TIM handle.  
**Return value:**None
  - **`__HAL_TIM_SetAutoreload`**  
**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_GetAutoreload`**  
**Description:** Gets the TIM Autoreload Register value on runtime.  
**Parameters:** `__HANDLE__`: TIM handle.  
**Return value:**None
  - **`__HAL_TIM_SetClockDivision`**  
**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_GetClockDivision`**  
**Description:** Gets the TIM Clock Division value on runtime.  
**Parameters:** `__HANDLE__`: TIM handle.  
**Return value:**None
  - **`__HAL_TIM_SetICPrescaler`**  
**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\_GetICPrescaler**

**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:

#### ***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 Filter Value***

- **IS\_TIM\_IC\_FILTER**

#### ***TIM Input Capture Polarity***

- **TIM\_ICPOLARITY\_RISING**
- **TIM\_ICPOLARITY\_FALLING**
- **TIM\_ICPOLARITY\_BOTHEDGE**
- **IS\_TIM\_IC\_POLARITY**

#### ***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
- **IS\_TIM\_IC\_PRESCALER**

#### ***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

- **IS\_TIM\_IC\_SELECTION**

**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**
- **IS\_TIM\_LOCK\_LEVEL**

**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**
- **IS\_TIM\_TRGO\_SOURCE**

**TIM Master Slave Mode**

- **TIM\_MASTERSLAVEMODE\_ENABLE**
- **TIM\_MASTERSLAVEMODE\_DISABLE**
- **IS\_TIM\_MSM\_STATE**

**TIM One Pulse Mode**

- **TIM\_OPMODE\_SINGLE**
- **TIM\_OPMODE\_REPETITIVE**
- **IS\_TIM\_OPM\_MODE**

**TIM OSSI Off State Selection for Idle mode state**

- **TIM\_OSSI\_ENABLE**
- **TIM\_OSSI\_DISABLE**
- **IS\_TIM\_OSSI\_STATE**

**TIM OSSR Off State Selection for Run mode state**

- **TIM\_OSSR\_ENABLE**

- **TIM\_OSSR\_DISABLE**
- **IS\_TIM\_OSSR\_STATE**

***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**
- **IS\_TIM\_PWM\_MODE**
- **IS\_TIM\_OC\_MODE**

***TIM Output Compare Idle State***

- **TIM\_OCIDLESTATE\_SET**
- **TIM\_OCIDLESTATE\_RESET**
- **IS\_TIM\_OCIDLE\_STATE**

***TIM Output Compare Polarity***

- **TIM\_OCPOLARITY\_HIGH**
- **TIM\_OCPOLARITY\_LOW**
- **IS\_TIM\_OC\_POLARITY**

***TIM Output Compare State***

- **TIM\_OUTPUTSTATE\_DISABLE**
- **TIM\_OUTPUTSTATE\_ENABLE**
- **IS\_TIM\_OUTPUT\_STATE**

***TIM Output Fast State***

- **TIM\_OCFAST\_DISABLE**
- **TIM\_OCFAST\_ENABLE**
- **IS\_TIM\_FAST\_STATE**

***TIM Private Constants***

- **CCER\_CCxE\_MASK**

***TIM Slave Mode***

- **TIM\_SLAVERESET\_DISABLE**
- **TIM\_SLAVERESET\_RESET**
- **TIM\_SLAVERESET\_GATED**
- **TIM\_SLAVERESET\_TRIGGER**
- **TIM\_SLAVERESET\_EXTERNAL1**
- **IS\_TIM\_SLAVE\_MODE**

***TIM TI1 Selection***

- **TIM\_TI1SELECTION\_CH1**
- **TIM\_TI1SELECTION\_XORCOMBINATION**
- **IS\_TIM\_TI1SELECTION**

***TIM Trigger Filter***

- **IS\_TIM\_TRIGGERFILTER**

***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
- **IS\_TIM\_TRIGGERPOLARITY**

***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.
- **IS\_TIM\_TRIGGERPRESCALER**

***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**
- **IS\_TIM\_TRIGGER\_SELECTION**
- **IS\_TIM\_INTERNAL\_TRIGGER\_SELECTION**
- **IS\_TIM\_INTERNAL\_TRIGGEREVENT\_SELECTION**

## 42 HAL TIM Extension Driver

### 42.1 TIMEx Firmware driver registers structures

#### 42.1.1 TIM\_MasterConfigTypeDef

*TIM\_MasterConfigTypeDef* is defined in the `stm32l1xx_hal_tim_ex.h`

##### 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`](#)

### 42.2 TIMEx Firmware driver API description

The following section lists the various functions of the TIMEx library.

#### 42.2.1 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

#### 42.2.2 Peripheral Control functions

This section provides functions allowing to:

- Configure Master synchronization.
- Configure timer remapping capabilities.
- [`HAL\_TIMEx\_MasterConfigSynchronization\(\)`](#)
- [`HAL\_TIMEx\_RemapConfig\(\)`](#)

#### 42.2.3 HAL\_TIMEx\_MasterConfigSynchronization

|               |                                                                                                                                                                               |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name | <code>HAL_StatusTypeDef<br/>HAL_TIMEx_MasterConfigSynchronization (</code><br><a href="#"><code>TIM_HandleTypeDef * htim, TIM_MasterConfigTypeDef * sMasterConfig)</code></a> |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                      |                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Configures the TIM in master mode.                                                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle.</li> <li>• <b>sMasterConfig</b> : pointer to a <b>TIM_MasterConfigTypeDef</b> structure that contains the selected trigger output (TRGO) and the Master/Slave mode.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                         |

#### 42.2.4 HAL\_TIMEx\_RemapConfig

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_TIMEx_RemapConfig (</b><br><b>TIM_HandleTypeDef * htim, uint32_t Remap)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Function Description | Configures the TIM2/TIM3/TIM9/TIM10/TIM11 Remapping input capabilities.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>htim</b> : TIM handle.</li> <li>• <b>Remap</b> : specifies the TIM remapping source. This parameter is a combination of the following values depending on TIM instance.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Notes                | <ul style="list-style-type: none"> <li>• For TIM2, the parameter can have the following values:<br/>TIM_TIM2_ITR1_TIM10_OC: TIM2 ITR1 input is connected to TIM10 OC<br/>TIM_TIM2_ITR1_TIM5_TGO: TIM2 ITR1 input is connected to TIM5 TGO</li> <li>• For TIM3, the parameter can have the following values:<br/>TIM_TIM3_ITR2_TIM11_OC: TIM3 ITR2 input is connected to TIM11 OC<br/>TIM_TIM3_ITR2_TIM5_TGO: TIM3 ITR2 input is connected to TIM5 TGO</li> <li>• For TIM9, the parameter is a combination of 2 fields (field1   field2): <ul style="list-style-type: none"> <li>• For TIM9, the field1 can have the following values:<br/>TIM_TIM9_ITR1_TIM3_TGO: TIM9 ITR1 input is connected to TIM3 TGO<br/>TIM_TIM9_ITR1_TS: TIM9 ITR1 input is connected to touch sensing I/O</li> <li>• For TIM9, the field2 can have the following values:<br/>TIM_TIM9_GPIO: TIM9 Channel1 is connected to GPIO<br/>TIM_TIM9_LSE: TIM9 Channel1 is connected to LSE internal clock<br/>TIM_TIM9_GPIO1: TIM9 Channel1 is connected to GPIO<br/>TIM_TIM9_GPIO2: TIM9 Channel1 is connected to GPIO</li> </ul> </li> <li>• For TIM10, the parameter is a combination of 3 fields (field1   field2   field3): <ul style="list-style-type: none"> <li>• For TIM10, the field1 can have the following values:<br/>TIM_TIM10_TI1RMP: TIM10 Channel 1 depends on TI1_RMP<br/>TIM_TIM10_RI: TIM10 Channel 1 is connected to</li> </ul> </li> </ul> |

## 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):
- 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

## 42.3 TIME<sub>x</sub> Firmware driver defines

### 42.3.1 TIME<sub>x</sub>

TIME<sub>x</sub>

#### *TIME<sub>x</sub> 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**

## 43 HAL UART Generic Driver

### 43.1 UART Firmware driver registers structures

#### 43.1.1 **UART\_InitTypeDef**

*UART\_InitTypeDef* is defined in the `stm32l1xx_hal_uart.h`

##### 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:
  - IntegerDivider = ((PCLKx) / (8 \* (OVR8+1) \* (uart->Init.BaudRate)))
  - FractionalDivider = ((IntegerDivider - ((uint32\_t) IntegerDivider)) \* 8 \* (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\*\*](#)

#### 43.1.2 **UART\_HandleTypeDef**

*UART\_HandleTypeDef* is defined in the `stm32l1xx_hal_uart.h`

##### 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 HAL_UART_ErrorTypeDef 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 HAL_UART_ErrorTypeDef UART_HandleTypeDef::ErrorCode` UART Error code

## 43.2 UART Firmware driver API description

The following section lists the various functions of the UART library.

### 43.2.1 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 these 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 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 UART 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 Baud Rate, Word Length, Stop Bit, Parity, Hardware flow control and Mode(Receiver/Transmitter) in the huart 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 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 using HAL\_UART\_Receive\_IT()
- 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

### 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



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

## 43.2.2 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: If the parity is enabled, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit. Depending on the frame

length defined by the M bit (8-bits or 9-bits), the possible UART frame formats are as listed in the following table:

- Hardware flow control
- Receiver/transmitter modes
- Over Sampling Methode

**Table 24: Frame formats**

| M bit | PCE bit | UART frame                 |
|-------|---------|----------------------------|
| 0     | 0       | SB   8 bit data   STB      |
| 0     | 1       | SB   7 bit data   PB   STB |
| 1     | 0       | SB   9 bit data   STB      |
| 1     | 1       | SB   8 bit data   PB   STB |

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)).

- [\*\*HAL\\_UART\\_Init\(\)\*\*](#)
- [\*\*HAL\\_HalfDuplex\\_Init\(\)\*\*](#)
- [\*\*HAL\\_LIN\\_Init\(\)\*\*](#)
- [\*\*HAL\\_MultiProcessor\\_Init\(\)\*\*](#)
- [\*\*HAL\\_UART\\_DeInit\(\)\*\*](#)
- [\*\*HAL\\_UART\\_MspInit\(\)\*\*](#)
- [\*\*HAL\\_UART\\_MspDeInit\(\)\*\*](#)

### 43.2.3 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.

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

#### 43.2.4 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
- [\*\*HAL\\_LIN\\_SendBreak\(\)\*\*](#)
- [\*\*HAL\\_MultiProcessor\\_EnterMuteMode\(\)\*\*](#)
- [\*\*HAL\\_MultiProcessor\\_ExitMuteMode\(\)\*\*](#)
- [\*\*HAL\\_HalfDuplex\\_EnableTransmitter\(\)\*\*](#)
- [\*\*HAL\\_HalfDuplex\\_EnableReceiver\(\)\*\*](#)

#### 43.2.5 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.
- ***HAL\_UART\_GetState()***
- ***HAL\_UART\_GetError()***

### 43.2.6 HAL\_UART\_Init

|                      |                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_Init ( <i>UART_HandleTypeDef</i> * <i>huart</i>)</b>                                                                                                                    |
| Function Description | Initializes the UART mode according to the specified parameters in the <i>UART_InitTypeDef</i> and create the associated handle.                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b><i>huart</i></b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                             |

### 43.2.7 HAL\_HalfDuplex\_Init

|                      |                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_HalfDuplex_Init ( <i>UART_HandleTypeDef</i> * <i>huart</i>)</b>                                                                                                              |
| Function Description | Initializes the half-duplex mode according to the specified parameters in the <i>UART_InitTypeDef</i> and create the associated handle.                                                               |
| Parameters           | <ul style="list-style-type: none"> <li>• <b><i>huart</i></b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                             |

### 43.2.8 HAL\_LIN\_Init

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_LIN_Init ( <i>UART_HandleTypeDef</i> * <i>huart</i>, <i>uint32_t</i> <i>BreakDetectLength</i>)</b>                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Function Description | Initializes the LIN mode according to the specified parameters in the <i>UART_InitTypeDef</i> and create the associated handle.                                                                                                                                                                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> <li>• <b>BreakDetectLength</b> : Specifies the LIN break detection length. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b><i>UART_LINBREAKDETECTLENGTH_10B</i></b> : 10-bit break detection</li> <li>– <b><i>UART_LINBREAKDETECTLENGTH_11B</i></b> : 11-bit break detection</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

### 43.2.9 HAL\_MultiProcessor\_Init

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_MultiProcessor_Init ( <i>UART_HandleTypeDef</i> * <i>huart</i>, <i>uint8_t</i> <i>Address</i>, <i>uint32_t</i> <i>WakeUpMethod</i>)</b>                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Function Description | Initializes the Multi-Processor mode according to the specified parameters in the <i>UART_InitTypeDef</i> and create the associated handle.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> <li>• <b>Address</b> : UART node address</li> <li>• <b>WakeUpMethod</b> : specifies the UART wakeup method. This parameter can be one of the following values: <ul style="list-style-type: none"> <li>– <b><i>UART_WAKEUPMETHOD_IDLELINE</i></b> : Wakeup by an idle line detection</li> <li>– <b><i>UART_WAKEUPMETHOD_ADDRESSMARK</i></b> : Wakeup by an address mark</li> </ul> </li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

### 43.2.10 HAL\_UART\_DelInit

|                      |                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_UART_DelInit (</code><br><code>  UART_HandleTypeDef * huart)</code>                                                                                                    |
| Function Description | Deinitializes the UART peripheral.                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                            |

### 43.2.11 HAL\_UART\_MspInit

|                      |                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_UART_MspInit (</code><br><code>  UART_HandleTypeDef * huart)</code>                                                                                                                 |
| Function Description | UART MSP Init.                                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                            |

### 43.2.12 HAL\_UART\_MspDeInit

|                      |                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_UART_MspDeInit (</code><br><code>  UART_HandleTypeDef * huart)</code>                                                                                                               |
| Function Description | UART MSP DeInit.                                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                            |

## Notes

- None.

### 43.2.13 HAL\_UART\_Transmit

|                      |                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_Transmit (</b><br><b>UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size,</b><br><b>uint32_t Timeout)</b>                                                                                                                                                                                                    |
| Function Description | Sends an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <b>UART_HandleTypeDef</b> structure that contains the configuration information for the specified <b>UART</b> module.</li><li>• <b>pData</b> : Pointer to data buffer</li><li>• <b>Size</b> : Amount of data to be sent</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                               |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                           |

### 43.2.14 HAL\_UART\_Receive

|                      |                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_Receive (</b><br><b>UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size,</b><br><b>uint32_t Timeout)</b>                                                                                                                                                                                                         |
| Function Description | Receives an amount of data in blocking mode.                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <b>UART_HandleTypeDef</b> structure that contains the configuration information for the specified <b>UART</b> module.</li><li>• <b>pData</b> : Pointer to data buffer</li><li>• <b>Size</b> : Amount of data to be received</li><li>• <b>Timeout</b> : Timeout duration</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                               |

### 43.2.15 HAL\_UART\_Transmit\_IT

|                      |                                                                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_Transmit_IT (</b><br><b>  UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                                                  |
| Function Description | Sends an amount of data in non blocking mode.                                                                                                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                |

### 43.2.16 HAL\_UART\_Receive\_IT

|                      |                                                                                                                                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_Receive_IT (</b><br><b>  UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                                                       |
| Function Description | Receives an amount of data in non blocking mode.                                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                    |

### 43.2.17 HAL\_UART\_Transmit\_DMA

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_Transmit_DMA (</b><br><b>  UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)</b>                                                                     |
| Function Description | Sends an amount of data in non blocking mode.                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART</li> </ul> |

## Return values

module.

- **pData** : Pointer to data buffer
- **Size** : Amount of data to be sent
- **HAL status**
- None.

## Notes

**43.2.18 HAL\_UART\_Receive\_DMA**

|                      |                                                                                                                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_Receive_DMA (</b><br><b><i>UART_HandleTypeDef</i> * huart, uint8_t * pData, uint16_t Size)</b>                                                                                                                                                                           |
| Function Description | Receives an amount of data in non blocking mode.                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> <li>• <b>pData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                    |
| Notes                | • When the UART parity is enabled (PCE = 1), the received data contain the parity bit (MSB position)                                                                                                                                                                                                   |

**43.2.19 HAL\_UART\_DMAPause**

|                      |                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_DMAPause (</b><br><b><i>UART_HandleTypeDef</i> * huart)</b>                                                                                                      |
| Function Description | Pauses the DMA Transfer.                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                            |
| Notes                | • None.                                                                                                                                                                                        |

### 43.2.20 HAL\_UART\_DMAResume

|                      |                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_DMAResume (</b><br><i>UART_HandleTypeDef * huart)</i>                                                                                                            |
| Function Description | Resumes the DMA Transfer.                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                      |

### 43.2.21 HAL\_UART\_DMAStop

|                      |                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_UART_DMAStop (</b><br><i>UART_HandleTypeDef * huart)</i>                                                                                                              |
| Function Description | Stops the DMA Transfer.                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                          |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                      |

### 43.2.22 HAL\_UART\_IRQHandler

|                      |                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_UART_IRQHandler (</b><br><i>UART_HandleTypeDef * huart)</i>                                                                                                                        |
| Function Description | This function handles UART interrupt request.                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                      |

### 43.2.23 HAL\_UART\_TxCpltCallback

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_UART_TxCpltCallback ( <i>UART_HandleTypeDef</i> * <i>huart</i>)</b>                                                                                                              |
| Function Description | Tx Transfer completed callbacks.                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |

### 43.2.24 HAL\_UART\_TxHalfCpltCallback

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_UART_TxHalfCpltCallback ( <i>UART_HandleTypeDef</i> * <i>huart</i>)</b>                                                                                                          |
| Function Description | Tx Half Transfer completed callbacks.                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |

### 43.2.25 HAL\_UART\_RxCpltCallback

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_UART_RxCpltCallback ( <i>UART_HandleTypeDef</i> * <i>huart</i>)</b> |
| Function Description | Rx Transfer completed callbacks.                                                |

---

|               |                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                            |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                            |

### 43.2.26 HAL\_UART\_RxHalfCpltCallback

|                      |                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_UART_RxHalfCpltCallback ( <i>UART_HandleTypeDef</i> * <b>huart</b>)</code>                                                                                                            |
| Function Description | Rx Half Transfer completed callbacks.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                            |

### 43.2.27 HAL\_UART\_ErrorCallback

|                      |                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_UART_ErrorCallback ( <i>UART_HandleTypeDef</i> * <b>huart</b>)</code>                                                                                                                 |
| Function Description | UART error callbacks.                                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <code>UART_HandleTypeDef</code> structure that contains the configuration information for the specified UART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                            |

### 43.2.28 HAL\_LIN\_SendBreak

|                      |                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_LIN_SendBreak (</b><br><b>UART_HandleTypeDef * huart)</b>                                                                                                                  |
| Function Description | Transmits break characters.                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <b>UART_HandleTypeDef</b> structure that contains the configuration information for the specified <b>UART</b> module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                             |

### 43.2.29 HAL\_MultiProcessor\_EnterMuteMode

|                      |                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_MultiProcessor_EnterMuteMode (</b><br><b>UART_HandleTypeDef * huart)</b>                                                                                                   |
| Function Description | Enters the <b>UART</b> in mute mode.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <b>UART_HandleTypeDef</b> structure that contains the configuration information for the specified <b>UART</b> module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                             |

### 43.2.30 HAL\_MultiProcessor\_ExitMuteMode

|                      |                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_MultiProcessor_ExitMuteMode (</b><br><b>UART_HandleTypeDef * huart)</b>                                                                                                    |
| Function Description | Exits the <b>UART</b> mute mode: wake up software.                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <b>UART_HandleTypeDef</b> structure that contains the configuration information for the specified <b>UART</b> module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                 |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                             |

### 43.2.31 HAL\_HalfDuplex\_EnableTransmitter

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_HalfDuplex_EnableTransmitter (</b><br><b><i>UART_HandleTypeDef * huart</i></b> )                                                                                    |
| Function Description | Enables the UART transmitter and disables the UART receiver.                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <b>UART_HandleTypeDef</b> structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                          |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |

### 43.2.32 HAL\_HalfDuplex\_EnableReceiver

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_HalfDuplex_EnableReceiver (</b><br><b><i>UART_HandleTypeDef * huart</i></b> )                                                                                       |
| Function Description | Enables the UART receiver and disables the UART transmitter.                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <b>UART_HandleTypeDef</b> structure that contains the configuration information for the specified UART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                          |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                      |

### 43.2.33 HAL\_UART\_GetState

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_UART_StateTypeDef HAL_UART_GetState (</b><br><b><i>UART_HandleTypeDef * huart</i></b> )                                                                                               |
| Function Description | Returns the UART state.                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>huart</b> : Pointer to a <b>UART_HandleTypeDef</b> structure that contains the configuration information for the specified UART module.</li></ul> |

---

|               |                                                                      |
|---------------|----------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL state</b></li> </ul> |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>            |

### 43.2.34 HAL\_UART\_GetError

|                      |                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>uint32_t HAL_UART_GetError ( <i>UART_HandleTypeDef</i> * <i>huart</i>)</b>                                                                                                           |
| Function Description | Return the UART error code.                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>huart</b> : Pointer to a <i>UART_HandleTypeDef</i> structure that contains the configuration information for the specified UART.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>UART Error Code</b></li> </ul>                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                               |

## 43.3 UART Firmware driver defines

### 43.3.1 UART

UART

#### *UART Exported Macros*

- **\_HAL\_UART\_RESET\_HANDLE\_STATE**  
**Description:** Reset UART handle state.  
**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\_FLUSH\_DRREGISTER**  
**Description:** Flushes the UART DR register.  
**Parameters:** *HANDLE*: specifies the UART Handle.
- **\_HAL\_UART\_GET\_FLAG**  
**Description:** Checks whether the specified UART flag is set or not.  
**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). *FLAG*: specifies the flag to check. This parameter can be one of the following values: *UART\_FLAG\_CTS*: CTS Change flag (not available for USART4 and USART5) *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:** Clears the specified UART pending flag.  
**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). 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:
  - **HAL\_UART\_CLEAR\_PEFLAG**  
**Description:** Clear the UART PE pending flag.  
**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\_CLEAR\_FEFLAG**  
**Description:** Clear the UART FE pending flag.  
**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\_CLEAR\_NEFLAG**  
**Description:** Clear the UART NE pending flag.  
**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\_CLEAR\_OREFLAG**  
**Description:** Clear the UART ORE pending flag.  
**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\_CLEAR\_IDLEFLAG**  
**Description:** Clear the UART IDLE pending flag.  
**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\_ENABLE\_IT**  
**Description:** Enables or disables the specified UART interrupt.  
**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). INTERRUPT: specifies the UART interrupt source to check. 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**

- **`_HAL_UART_GET_IT_SOURCE`**

**Description:** Checks whether the specified UART interrupt has occurred or not.  
**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). `_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  
`USART_IT_ERR`: Error interrupt

**Return value:** The new state of `_IT_` (TRUE or FALSE).

- **`_HAL_UART_ONEBIT_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_ONEBIT_DISABLE`**

- **`_HAL_UART_ENABLE`**

**Description:** Enable UART.

**Parameters:** `_HANDLE_`: specifies the UART Handle. The Handle Instance 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_DISABLE`**

**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`
- `IS_UART_HARDWARE_FLOW_CONTROL`

### ***UART interruptions flag mask***

- `UART_IT_MASK`

### ***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**
- **IS\_UART\_LIN\_BREAK\_DETECT\_LENGTH**

***UART Transfer Mode***

- **UART\_MODE\_RX**
- **UART\_MODE\_TX**
- **UART\_MODE\_TX\_RX**
- **IS\_UART\_MODE**

***UART Over Sampling***

- **UART\_OVERSAMPLING\_16**
- **UART\_OVERSAMPLING\_8**
- **IS\_UART\_OVERSAMPLING**

***UART Parity***

- **UART\_PARITY\_NONE**
- **UART\_PARITY\_EVEN**
- **UART\_PARITY\_ODD**
- **IS\_UART\_PARITY**

***UART Private Constants***

- **UART\_TIMEOUT\_VALUE**

***UART Private Macros***

- **UART\_DIV\_SAMPLING16**
- **UART\_DIVMANT\_SAMPLING16**
- **UART\_DIVFRAQ\_SAMPLING16**
- **UART\_BRR\_SAMPLING16**
- **UART\_DIV\_SAMPLING8**
- **UART\_DIVMANT\_SAMPLING8**
- **UART\_DIVFRAQ\_SAMPLING8**
- **UART\_BRR\_SAMPLING8**
- **IS\_UART\_BAUDRATE**

**Description:** Check UART Baud rate.

**Parameters:** `__BAUDRATE__`: Baudrate specified by the user. The maximum Baud Rate is derived from the maximum clock on APB (i.e. 32 MHz) divided by the smallest oversampling used on the USART (i.e. 8).

**Return value:** Test: result (TRUE or FALSE).

- **IS\_UART\_ADDRESS**

**Description:** Check UART Node Address.

**Parameters:** `__ADDRESS__`: UART Node address specified by the user. UART Node address is used in Multi processor communication for wakeup with address mark detection. This parameter must be a number between `Min_Data = 0` and `Max_Data = 15`.

**Return value:** Test: result (TRUE or FALSE).

***UART State***

- **UART\_STATE\_DISABLE**
- **UART\_STATE\_ENABLE**
- **IS\_UART\_STATE**

***UART Number of Stop Bits***

- **UART\_STOPBITS\_1**
- **UART\_STOPBITS\_2**
- **IS\_UART\_STOPBITS**

***UART Wakeup Functions***

- **UART\_WAKEUPMETHOD\_IDLELINE**
- **UART\_WAKEUPMETHOD\_ADDRESSMARK**
- **IS\_UART\_WAKEUPMETHOD**

***UART Word Length***

- **UART\_WORDLENGTH\_8B**
- **UART\_WORDLENGTH\_9B**
- **IS\_UART\_WORD\_LENGTH**

## 44 HAL USART Generic Driver

### 44.1 USART Firmware driver registers structures

#### 44.1.1 USART\_InitTypeDef

**USART\_InitTypeDef** is defined in the `stm32l1xx_hal_usart.h`

##### 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 wether 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\*\*](#)

#### 44.1.2 USART\_HandleTypeDef

**USART\_HandleTypeDef** is defined in the `stm32l1xx_hal_usart.h`

##### 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 HAL\_USART\_ErrorTypeDef 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 HAL\_USART\_ErrorTypeDef USART\_HandleTypeDef::ErrorCode*** USART Error code

## 44.2 USART Firmware driver API description

The following section lists the various functions of the USART library.

### 44.2.1 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 these 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.
3. Program the Baud Rate, Word Length, Stop Bit, Parity, Hardware flow control and Mode(Receiver/Transmitter) in the usart 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 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 using HAL\_USART\_Receive\_IT()
- 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

## 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



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

### 44.2.2 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: If the parity is enabled, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit. Depending on the frame length defined by the M bit (8-bits or 9-bits), the possible USART frame formats are as listed in the below table:

- USART polarity
- USART phase
- USART LastBit
- Receiver/transmitter modes

**Table 25: Frame formats**

| M bit | PCE bit | USART frame                |
|-------|---------|----------------------------|
| 0     | 0       | SB   8 bit data   STB      |
| 0     | 1       | SB   7 bit data   PB   STB |
| 1     | 0       | SB   9 bit data   STB      |
| 1     | 1       | SB   8 bit data   PB   STB |

The HAL\_USART\_Init() function follows the USART synchronous configuration procedure (details for the procedure are available in reference manual (RM0038)).

- [\*\*HAL\\_USART\\_Init\(\)\*\*](#)
- [\*\*HAL\\_USART\\_DeInit\(\)\*\*](#)
- [\*\*HAL\\_USART\\_MspInit\(\)\*\*](#)
- [\*\*HAL\\_USART\\_MspDeInit\(\)\*\*](#)

#### 44.2.3 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\_TxRx\_CpltCallback() 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()
  - ***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()***

#### 44.2.4 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.
- ***HAL\_USART\_GetState()***
- ***HAL\_USART\_GetError()***

#### 44.2.5 HAL\_USART\_Init

|                      |                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_Init (</b><br><b><i>USART_HandleTypeDef * husart</i></b> )                                   |
| 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"> <li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                      |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                  |

#### 44.2.6 HAL\_USART\_DeInit

|                      |                                                                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_DeInit ( USART_HandleTypeDef * husart)</b>                                                                                                                  |
| Function Description | Deinitializes the USART peripheral.                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                      |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                  |

#### 44.2.7 HAL\_USART\_MspInit

|                      |                                                                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_USART_MspInit ( USART_HandleTypeDef * husart)</b>                                                                                                                              |
| Function Description | USART MSP Init.                                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                  |

#### 44.2.8 HAL\_USART\_MspDeInit

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_USART_MspDelInit ( USART_HandleTypeDef *husart)</b>                                                                                                                          |
| Function Description | USART MSP DelInit.                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li><b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                  |

#### 44.2.9 HAL\_USART\_Transmit

|                      |                                                                                                                                                                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_Transmit ( USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                                       |
| Function Description | Simplex Send an amount of data in blocking mode.                                                                                                                                                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"> <li><b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> <li><b>pTxData</b> : Pointer to data buffer</li> <li><b>Size</b> : Amount of data to be sent</li> <li><b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li><b>HAL status</b></li> </ul>                                                                                                                                                                                                                                                                   |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                                                                                                                                                               |

#### 44.2.10 HAL\_USART\_Receive

|                      |                                                                                                                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_Receive ( USART_HandleTypeDef * husart, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                 |
| Function Description | Full-Duplex Receive an amount of data in blocking mode.                                                                                                                                                                                                                                        |
| Parameters           | <ul style="list-style-type: none"> <li><b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> <li><b>pRxData</b> : Pointer to data buffer</li> <li><b>Size</b> : Amount of data to be received</li> </ul> |

- **Timeout** : Timeout duration
  - **HAL status**
  - None.
- Return values
- Notes

#### 44.2.11 HAL\_USART\_TransmitReceive

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_TransmitReceive (</b><br><b>USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)</b>                                                                                                                                                                                                                                                   |
| Function Description | Full-Duplex Send receive an amount of data in full-duplex mode (blocking mode).                                                                                                                                                                                                                                                                                                                                       |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> <li>• <b>pTxData</b> : Pointer to data transmitted buffer</li> <li>• <b>pRxData</b> : Pointer to data received buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> <li>• <b>Timeout</b> : Timeout duration</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                                                                   |
| Notes                | • None.                                                                                                                                                                                                                                                                                                                                                                                                               |

#### 44.2.12 HAL\_USART\_Transmit\_IT

|                      |                                                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_Transmit_IT (</b><br><b>USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t Size)</b>                                                                                                                                                                       |
| Function Description | Simplex Send an amount of data in non-blocking mode.                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> <li>• <b>pTxData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | • <b>HAL status</b>                                                                                                                                                                                                                                                                              |
| Notes                | • The USART errors are not managed to avoid the overrun error.                                                                                                                                                                                                                                   |

#### 44.2.13 HAL\_USART\_Receive\_IT

|                      |                                                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_USART_Receive_IT (<br/>    <b>USART_HandleTypeDef</b> * <b>husart</b>, uint8_t * <b>pRxData</b>, uint16_t<br/>    <b>Size</b>)</code>                                                                                                                                |
| Function Description | Simplex Receive an amount of data in non-blocking mode.                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li><li>• <b>pRxData</b> : Pointer to data buffer</li><li>• <b>Size</b> : Amount of data to be received</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                          |

#### 44.2.14 HAL\_USART\_TransmitReceive\_IT

|                      |                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_USART_TransmitReceive_IT (<br/>    <b>USART_HandleTypeDef</b> * <b>husart</b>, uint8_t * <b>pTxData</b>, uint8_t *<br/>    <b>pRxData</b>, uint16_t <b>Size</b>)</code>                                                                                                                                                                     |
| Function Description | Full-Duplex Send receive an amount of data in full-duplex mode (non-blocking).                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li><li>• <b>pTxData</b> : Pointer to data transmitted buffer</li><li>• <b>pRxData</b> : Pointer to data received buffer</li><li>• <b>Size</b> : Amount of data to be received</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                                                                                                                                                                                                     |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                                                                                                                                                                                                 |

#### 44.2.15 HAL\_USART\_Transmit\_DMA

|                      |                                                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_USART_Transmit_DMA (</code><br><code>  USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t</code><br><code>  Size)</code>                                                                                                                                      |
| Function Description | Simplex Send an amount of data in non-blocking mode.                                                                                                                                                                                                                                             |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> <li>• <b>pTxData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be sent</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                            |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                        |

#### 44.2.16 HAL\_USART\_Receive\_DMA

|                      |                                                                                                                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_USART_Receive_DMA (</code><br><code>  USART_HandleTypeDef * husart, uint8_t * pRxData, uint16_t</code><br><code>  Size)</code>                                                                                                                                           |
| Function Description | Full-Duplex Receive an amount of data in non-blocking mode.                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> <li>• <b>pRxData</b> : Pointer to data buffer</li> <li>• <b>Size</b> : Amount of data to be received</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• The USART DMA transmit channel must be configured in order to generate the clock for the slave.</li> <li>• When the USART parity is enabled (PCE = 1) the data received contain the parity bit.</li> </ul>                                                  |

#### 44.2.17 HAL\_USART\_TransmitReceive\_DMA

|                      |                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_TransmitReceive_DMA (</b><br><b>USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)</b>                                                                                                                                                                                                                |
| Function Description | Full-Duplex Transmit Receive an amount of data in non-blocking mode.                                                                                                                                                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li><b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> <li><b>pTxData</b> : Pointer to data transmitted buffer</li> <li><b>pRxData</b> : Pointer to data received buffer</li> <li><b>Size</b> : Amount of data to be received</li> </ul> |
| Return values        | <b>HAL status</b>                                                                                                                                                                                                                                                                                                                                                    |
| Notes                | <ul style="list-style-type: none"> <li>When the USART parity is enabled (PCE = 1) the data received contain the parity bit.</li> </ul>                                                                                                                                                                                                                               |

#### 44.2.18 HAL\_USART\_DMAPause

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_DMAPause (</b><br><b>USART_HandleTypeDef * husart)</b>                                                                                                    |
| Function Description | Pauses the DMA Transfer.                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"> <li><b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> </ul> |
| Return values        | <b>HAL status</b>                                                                                                                                                                        |
| Notes                | <ul style="list-style-type: none"> <li>None.</li> </ul>                                                                                                                                  |

#### 44.2.19 HAL\_USART\_DMAResume

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_DMAResume (</b><br><b>USART_HandleTypeDef * husart)</b>                                                                                                   |
| Function Description | Resumes the DMA Transfer.                                                                                                                                                                |
| Parameters           | <ul style="list-style-type: none"> <li><b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li> </ul> |

|               |                                                                     |
|---------------|---------------------------------------------------------------------|
| Return values | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul> |
| Notes         | <ul style="list-style-type: none"><li>• None.</li></ul>             |

#### 44.2.20 HAL\_USART\_DMASStop

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_USART_DMASStop ( <i>USART_HandleTypeDef</i> * <i>husart</i>)</b>                                                                                                |
| Function Description | Stops the DMA Transfer.                                                                                                                                                                  |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                      |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

#### 44.2.21 HAL\_USART\_IRQHandler

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_USART_IRQHandler ( <i>USART_HandleTypeDef</i> * <i>husart</i>)</b>                                                                                                           |
| Function Description | This function handles USART interrupt request.                                                                                                                                           |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

#### 44.2.22 HAL\_USART\_TxCpltCallback

|               |                                                                     |
|---------------|---------------------------------------------------------------------|
| Function Name | <b>void HAL_USART_TxCpltCallback ( <i>USART_HandleTypeDef</i> *</b> |
|---------------|---------------------------------------------------------------------|

**husart)**

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Description | Tx Transfer completed callbacks.                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

#### 44.2.23 HAL\_USART\_TxHalfCpltCallback

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_USART_TxHalfCpltCallback ( USART_HandleTypeDef * husart)</b>                                                                                                                 |
| Function Description | Tx Half Transfer completed callbacks.                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

#### 44.2.24 HAL\_USART\_RxCpltCallback

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_USART_RxCpltCallback ( USART_HandleTypeDef * husart)</b>                                                                                                                     |
| Function Description | Rx Transfer completed callbacks.                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

**44.2.25 HAL\_USART\_RxHalfCpltCallback**

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_USART_RxHalfCpltCallback ( <i>USART_HandleTypeDef</i> * <i>husart</i>)</b>                                                                                                   |
| Function Description | Rx Half Transfer completed callbacks.                                                                                                                                                    |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

**44.2.26 HAL\_USART\_TxRxCpltCallback**

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_USART_TxRxCpltCallback ( <i>USART_HandleTypeDef</i> * <i>husart</i>)</b>                                                                                                     |
| Function Description | Tx/Rx Transfers completed callback for the non-blocking process.                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

**44.2.27 HAL\_USART\_ErrorCallback**

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_USART_ErrorCallback ( <i>USART_HandleTypeDef</i> * <i>husart</i>)</b>                                                                                                        |
| Function Description | USART error callbacks.                                                                                                                                                                   |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

## Notes

- None.

#### 44.2.28 HAL\_USART\_GetState

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_USART_StateTypeDef HAL_USART_GetState (</code><br><code>USART_HandleTypeDef * husart)</code>                                                                                   |
| Function Description | Returns the USART state.                                                                                                                                                                 |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : Pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                                                                                                       |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                                  |

#### 44.2.29 HAL\_USART\_GetError

|                      |                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>uint32_t HAL_USART_GetError (</code><br><code>USART_HandleTypeDef * husart)</code>                                                                                          |
| Function Description | Return the USART error code.                                                                                                                                                      |
| Parameters           | <ul style="list-style-type: none"><li>• <b>husart</b> : pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>USART Error Code</b></li></ul>                                                                                                         |

### 44.3 USART Firmware driver defines

#### 44.3.1 USART

USART

*USART Clock*

- **USART\_CLOCK\_DISABLED**

- **USART\_CLOCK\_ENABLED**
- **IS\_USART\_CLOCK**

**USART Clock Phase**

- **USART\_PHASE\_1EDGE**
- **USART\_PHASE\_2EDGE**
- **IS\_USART\_PHASE**

**USART Clock Polarity**

- **USART\_POLARITY\_LOW**
- **USART\_POLARITY\_HIGH**
- **IS\_USART\_POLARITY**

**USART Exported Macros**

- **\_HAL\_USART\_RESET\_HANDLE\_STATE**

**Description:** Reset USART handle state.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral.

**Return value:**None:

- **\_HAL\_USART\_GET\_FLAG**

**Description:** Checks whether the specified USART flag is set or not.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral. `_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:** Clears the specified USART pending flags.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral. `_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:

- **\_HAL\_USART\_CLEAR\_PEFLAG**

**Description:** Clear the USART PE pending flag.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral.

**Return value:**None:

- **\_HAL\_USART\_CLEAR\_FEFLAG**

**Description:** Clear the USART FE pending flag.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral.

**Return value:**None:

- **\_HAL\_USART\_CLEAR\_NEFLAG**

**Description:** Clear the USART NE pending flag.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral.

**Return value:**None:

- **\_HAL\_USART\_CLEAR\_OREFLAG**

**Description:** Clear the USART ORE pending flag.



**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral.

**Return value:**None:

- `_HAL_USART_CLEAR_IDLEFLAG`

**Description:** Clear the USART IDLE pending flag.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral.

**Return value:**None:

- `_HAL_USART_ENABLE_IT`

**Description:** Enables or disables the specified Usart interrupts.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral. `_INTERRUPT_`:

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\_PE: Parity Error interrupt USART\_IT\_ERR: Error interrupt(Frame error, noise error, overrun error)

**Return value:**None:

- `_HAL_USART_DISABLE_IT`

- `_HAL_USART_GET_IT_SOURCE`

**Description:** Checks whether the specified Usart interrupt has occurred or not.

**Parameters:** `_HANDLE_`: specifies the USART Handle. This parameter can be USARTx where x: 1, 2 or 3 to select the USART peripheral. `_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_ENABLE`

**Description:** Enable USART.

**Parameters:** `_HANDLE_`: specifies the USART Handle. The Handle Instance can be USARTx where x: 1, 2, 3 to select the USART peripheral

**Return value:**None:

- `_HAL_USART_DISABLE`

**Description:** Disable USART.

**Parameters:** `_HANDLE_`: specifies the USART Handle. The Handle Instance can be USARTx where x: 1, 2, 3 to select the USART peripheral

**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 interruptions flag mask**

- **USART\_IT\_MASK**

***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**
- **IS\_USART\_LASTBIT**

***USART Mode***

- **USART\_MODE\_RX**
- **USART\_MODE\_TX**
- **USART\_MODE\_TX\_RX**
- **IS\_USART\_MODE**

***USART NACK State***

- **USARTNACK\_ENABLED**
- **USARTNACK\_DISABLED**
- **IS\_USART\_NACK\_STATE**

***USART Parity***

- **USART\_PARITY\_NONE**
- **USART\_PARITY\_EVEN**
- **USART\_PARITY\_ODD**
- **IS\_USART\_PARITY**

***USART Private Constants***

- **DUMMY\_DATA**
- **USART\_TIMEOUT\_VALUE**

***USART Private Macros***

- **USART\_DIV**
- **USART\_DIVMANT**
- **USART\_DIVFRAQ**
- **USART\_BRR**
- **IS\_USART\_BAUDRATE**

**Description:** Check USART Baud rate.

**Parameters:** **\_BAUDRATE\_**: Baudrate specified by the user. The maximum Baud Rate is derived from the maximum clock on APB (i.e. 32 MHz) divided by the smallest oversampling used on the USART (i.e. 8)

**Return value:** Test: result (TRUE or FALSE)

***USART Number of Stop Bits***

- **USART\_STOPBITS\_1**
- **USART\_STOPBITS\_0\_5**

- **USART\_STOPBITS\_2**
- **USART\_STOPBITS\_1\_5**
- **IS\_USART\_STOPBITS**

***USART Word Length***

- **USART\_WORDLENGTH\_8B**
- **USART\_WORDLENGTH\_9B**
- **IS\_USART\_WORD\_LENGTH**

## 45 HAL WWDG Generic Driver

### 45.1 WWDG Firmware driver registers structures

#### 45.1.1 WWDG\_InitTypeDef

*WWDG\_InitTypeDef* is defined in the `stm32l1xx_hal_wwdg.h`

##### Data Fields

- *uint32\_t Prescaler*
- *uint32\_t Window*
- *uint32\_t Counter*

##### 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 lower than Max\_Data = 0x80
- *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

#### 45.1.2 WWDG\_HandleTypeDefDef

*WWDG\_HandleTypeDefDef* is defined in the `stm32l1xx_hal_wwdg.h`

##### Data Fields

- *WWDG\_TypeDef \* Instance*
- *WWDG\_InitTypeDef Init*
- *HAL\_LockTypeDef Lock*
- *\_\_IO HAL\_WWDG\_StateTypeDef State*

##### Field Documentation

- *WWDG\_TypeDef\* WWDG\_HandleTypeDefDef::Instance* Register base address
- *WWDG\_InitTypeDef WWDG\_HandleTypeDefDef::Init* WWDG required parameters
- *HAL\_LockTypeDef WWDG\_HandleTypeDefDef::Lock* WWDG locking object
- *\_\_IO HAL\_WWDG\_StateTypeDef WWDG\_HandleTypeDefDef::State* WWDG communication state

### 45.2 WWDG Firmware driver API description

The following section lists the various functions of the WWDG library.

#### 45.2.1 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize the WWDG according to the specified parameters in the WWDG\_InitTypeDef and create the associated handle
- DeInitialize the WWDG peripheral
- Initialize the WWDG MSP
- DeInitialize the WWDG MSP
- [\*\*HAL\\_WWDG\\_Init\(\)\*\*](#)
- [\*\*HAL\\_WWDG\\_DeInit\(\)\*\*](#)
- [\*\*HAL\\_WWDG\\_MspInit\(\)\*\*](#)
- [\*\*HAL\\_WWDG\\_MspDeInit\(\)\*\*](#)
- [\*\*HAL\\_WWDG\\_WakeupCallback\(\)\*\*](#)

#### 45.2.2 IO operation functions

This section provides functions allowing to:

- Start the WWDG.
- Refresh the WWDG.
- Handle WWDG interrupt request.
- [\*\*HAL\\_WWDG\\_Start\(\)\*\*](#)
- [\*\*HAL\\_WWDG\\_Start\\_IT\(\)\*\*](#)
- [\*\*HAL\\_WWDG\\_Refresh\(\)\*\*](#)
- [\*\*HAL\\_WWDG\\_IRQHandler\(\)\*\*](#)
- [\*\*HAL\\_WWDG\\_WakeupCallback\(\)\*\*](#)

#### 45.2.3 Peripheral State functions

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

- [\*\*HAL\\_WWDG\\_GetState\(\)\*\*](#)

#### 45.2.4 HAL\_WWDG\_Init

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_WWDG_Init (</b><br><a href="#"><b>WWDG_HandleTypeDef * hwdg</b></a> )                                                                                       |
| Function Description | Initializes the WWDG according to the specified parameters in the WWDG_InitTypeDef and creates the associated handle.                                                                |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                              |

## 45.2.5 HAL\_WWDG\_DelInit

|                      |                                                                                                                                                                                        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_WWDG_DelInit ( <i>WWDG_HandleTypeDef</i> * hwdg)</b>                                                                                                          |
| Function Description | Deinitializes the WWDG peripheral.                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                  |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                              |

## 45.2.6 HAL\_WWDG\_MspInit

|                      |                                                                                                                                                                                        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_WWDG_MspInit ( <i>WWDG_HandleTypeDef</i> * hwdg)</b>                                                                                                                       |
| Function Description | Initializes the WWDG MSP.                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                              |

## 45.2.7 HAL\_WWDG\_MspDelInit

|                      |                                                                                                                                                                                        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_WWDG_MspDelInit ( <i>WWDG_HandleTypeDef</i> * hwdg)</b>                                                                                                                    |
| Function Description | Deinitializes the WWDG MSP.                                                                                                                                                            |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                              |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                              |

#### 45.2.8 HAL\_WWDG\_WakeupCallback

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_WWDG_WakeupCallback ( <i>WWDG_HandleTypeDef</i> * hwdg)</code>                                                                                                        |
| Function Description | Early Wakeup WWDG callback.                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                              |

#### 45.2.9 HAL\_WWDG\_Start

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_WWDG_Start ( <i>WWDG_HandleTypeDef</i> * hwdg)</code>                                                                                                    |
| Function Description | Starts the WWDG.                                                                                                                                                                     |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL status</b></li></ul>                                                                                                                  |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                              |

#### 45.2.10 HAL\_WWDG\_Start\_IT

|                      |                                                                                      |
|----------------------|--------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_StatusTypeDef HAL_WWDG_Start_IT ( <i>WWDG_HandleTypeDef</i> * hwdg)</code> |
| Function Description | Starts the WWDG with interrupt enabled.                                              |

|               |                                                                                                                                                                                         |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parameters    | <ul style="list-style-type: none"> <li>• <b>hwwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li> </ul> |
| Return values | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                   |
| Notes         | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                               |

#### 45.2.11 HAL\_WWDG\_Refresh

|                      |                                                                                                                                                                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>HAL_StatusTypeDef HAL_WWDG_Refresh ( <br/>WWDG_HandleTypeDef * hwwdg, uint32_t Counter)</b>                                                                                                                                                              |
| Function Description | Refreshes the WWDG.                                                                                                                                                                                                                                         |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hwwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li> <li>• <b>Counter</b> : value of counter to put in WWDG counter</li> </ul> |
| Return values        | <ul style="list-style-type: none"> <li>• <b>HAL status</b></li> </ul>                                                                                                                                                                                       |
| Notes                | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                   |

#### 45.2.12 HAL\_WWDG\_IRQHandler

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <b>void HAL_WWDG_IRQHandler ( <br/>WWDG_HandleTypeDef * hwwdg)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Function Description | Handles WWDG interrupt request.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"> <li>• <b>hwwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li> </ul>                                                                                                                                                                                                                                                                                                                                  |
| Return values        | <ul style="list-style-type: none"> <li>• None.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Notes                | <ul style="list-style-type: none"> <li>• 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 using __HAL_WWDG_ENABLE_IT() macro. 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.</li> </ul> |

#### 45.2.13 HAL\_WWDG\_WakeupCallback

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>void HAL_WWDG_WakeupCallback ( <i>WWDG_HandleTypeDef</i> * hwdg)</code>                                                                                                        |
| Function Description | Early Wakeup WWDG callback.                                                                                                                                                          |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                              |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                              |

#### 45.2.14 HAL\_WWDG\_GetState

|                      |                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Name        | <code>HAL_WWDG_StateTypeDef HAL_WWDG_GetState ( <i>WWDG_HandleTypeDef</i> * hwdg)</code>                                                                                             |
| Function Description | Returns the WWDG state.                                                                                                                                                              |
| Parameters           | <ul style="list-style-type: none"><li>• <b>hwdg</b> : pointer to a WWDG_HandleTypeDef structure that contains the configuration information for the specified WWDG module.</li></ul> |
| Return values        | <ul style="list-style-type: none"><li>• <b>HAL state</b></li></ul>                                                                                                                   |
| Notes                | <ul style="list-style-type: none"><li>• None.</li></ul>                                                                                                                              |

### 45.3 WWDG Firmware driver defines

#### 45.3.1 WWDG

WWDG

*WWDG BitAddress AliasRegion*

- **CFR\_BASE**

*WWDG Counter*

- **IS\_WWDG\_COUNTER**

**WWDG Exported Macros**

- **\_HAL\_WWDG\_RESET\_HANDLE\_STATE**

**Description:** Reset WWDG handle state.

**Parameters:** `_HANDLE_`: WWDG handle

**Return value:**None

- **\_HAL\_WWDG\_ENABLE**

**Description:** Enables the WWDG peripheral.

**Parameters:** `_HANDLE_`: WWDG handle

**Return value:**None

- **\_HAL\_WWDG\_GET\_FLAG**

**Description:** Gets the selected WWDG's flag status.

**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:** Clears 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\_ENABLE\_IT**

**Description:** Enables the WWDG early wakeup interrupt.

**Parameters:** `_INTERRUPT_`: specifies the interrupt to enable. This parameter can be one of the following values: `WWDG_IT_EWI`: Early wakeup interrupt

**Return value:**None

- **\_HAL\_WWDG\_CLEAR\_IT**

**Description:** Clear the WWDG's interrupt pending bits to clear the selected 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

**WWDG Flag definition**

- **WWDG\_FLAG\_EWIF**

Early wakeup interrupt flag

**WWDG Interrupt definition**

- **WWDG\_IT\_EWI**

**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

- **IS\_WWDG\_PRESCALER**

**WWDG Window**

- **IS\_WWDG\_WINDOW**

**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 .

**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 drivers 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\_MspDelinit() functions?**

These function are called within HAL\_PPP\_Init() and HAL\_PPP\_Delinit(), 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.

## 47 Revision history

Table 26: Document revision history

| Date        | Revision | Changes          |
|-------------|----------|------------------|
| 20-Nov-2014 | 1        | Initial release. |

**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.

© 2014 STMicroelectronics – All rights reserved