5 HAL System Driver

5.1 HAL Firmware driver API description

5.1.1 How to use this driver

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

The HAL contains two APIs' categories:

- Common HAL APIs
- Services HAL APIs

5.1.2 Initialization and de-initialization functions

This section provides functions allowing to:

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

This section contains the following APIs:

- ***HAL_Init()***
- ***HAL_DeInit()***
- ***HAL_MspInit()***
- ***HAL_MspDeInit()***
- ***HAL_InitTick()***

5.1.3 HAL Control functions

This section provides functions allowing to:

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

This section contains the following APIs:

- ***HAL_IncTick()***
- ***HAL_GetTick()***
- ***HAL_Delay()***
- ***HAL_SuspendTick()***
- ***HAL_ResumeTick()***
- ***HAL_GetHalVersion()***
- ***HAL_GetREVID()***
- ***HAL_GetDEVID()***
- ***HAL_GetUIDw0()***
- ***HAL_GetUIDw1()***
- ***HAL_GetUIDw2()***

5.1.4 HAL Debug functions

This section provides functions allowing to:

- Enable/Disable Debug module during SLEEP mode
- Enable/Disable Debug module during STOP0/STOP1/STOP2 modes
- Enable/Disable Debug module during STANDBY mode

This section contains the following APIs:

- ***HAL_DBGMCU_EnableDBGSleepMode()***
- ***HAL_DBGMCU_DisableDBGSleepMode()***
- ***HAL_DBGMCU_EnableDBGStopMode()***
- ***HAL_DBGMCU_DisableDBGStopMode()***
- ***HAL_DBGMCU_EnableDBGStandbyMode()***
- ***HAL_DBGMCU_DisableDBGStandbyMode()***

5.1.5 HAL SYSCFG configuration functions

This section provides functions allowing to:

- Start a hardware SRAM2 erase operation
- Enable/Disable the Internal FLASH Bank Swapping
- Configure the Voltage reference buffer
- Enable/Disable the Voltage reference buffer
- Enable/Disable the I/O analog switch voltage booster

This section contains the following APIs:

- ***HAL_SYSCFG_SRAM2Erase()***
- ***HAL_SYSCFG_EnableMemorySwappingBank()***
- ***HAL_SYSCFG_DisableMemorySwappingBank()***
- ***HAL_SYSCFG_VREFBUF_VoltageScalingConfig()***
- ***HAL_SYSCFG_VREFBUF_HighImpedanceConfig()***
- ***HAL_SYSCFG_VREFBUF_TrimmingConfig()***
- ***HAL_SYSCFG_EnableVREFBUF()***
- ***HAL_SYSCFG_DisableVREFBUF()***
- ***HAL_SYSCFG_EnableIOAnalogSwitchBooster()***
- ***HAL_SYSCFG_DisableIOAnalogSwitchBooster()***

5.1.6 Detailed description of functions

HAL_Init

Function name	**HAL_StatusTypeDef HAL_Init (void)**
Function description	Configure the Flash prefetch, the Instruction and Data caches, the time base source, NVIC and any required global low level hardware by calling the HAL_MspInit() callback function to be optionally defined in user file stm32l4xx_hal_msp.c.
Return values	• **HAL:** status
Notes	• HAL_Init() function is called at the beginning of program after reset and before the clock configuration. • In the default implementation the System Timer (Systick) is used as source of time base. The Systick configuration is based on MSI clock, as MSI is the clock used after a system Reset and the NVIC configuration is set to Priority group 4. Once done, time base tick starts incrementing: the tick variable counter is incremented each 1ms in the SysTick_Handler() interrupt handler.

HAL_DeInit

Function name	**HAL_StatusTypeDef HAL_DeInit (void)**
Function description	De-initialize common part of the HAL and stop the source of time base.
Return values	• **HAL:** status
Notes	• This function is optional.

HAL_MspInit

Function name	**void HAL_MspInit (void)**
Function description	Initialize the MSP.
Return values	• **None:**

HAL_MspDeInit

Function name	**void HAL_MspDeInit (void)**
Function description	DeInitialize the MSP.
Return values	• **None:**

HAL_InitTick

Function name	**HAL_StatusTypeDef HAL_InitTick (uint32_t TickPriority)**
Function description	This function configures the source of the time base: The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.
Parameters	• **TickPriority:** Tick interrupt priority.

Return values
- **HAL:** status

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

HAL_IncTick

Function name **void HAL_IncTick (void)**

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

Return values
- **None:**

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

HAL_Delay

Function name **void HAL_Delay (uint32_t Delay)**

Function description This function provides minimum delay (in milliseconds) based on variable incremented.

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

Return values
- **None:**

Notes
- In the default implementation , SysTick timer is the source of time base. It is used to generate interrupts at regular time intervals where uwTick is incremented.
- This function is declared as __weak to be overwritten in case of other implementations in user file.

HAL_GetTick

Function name **uint32_t HAL_GetTick (void)**

Function description Provide a tick value in millisecond.

Return values
- **tick:** value

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

HAL_SuspendTick

Function name **void HAL_SuspendTick (void)**

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

HAL_ResumeTick

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

HAL_GetHalVersion

Function name	**uint32_t HAL_GetHalVersion (void)**
Function description	Return the HAL revision.
Return values	• **version:** 0xXYZR (8bits for each decimal, R for RC)

HAL_GetREVID

Function name	**uint32_t HAL_GetREVID (void)**
Function description	Return the device revision identifier.
Return values	• **Device:** revision identifier

HAL_GetDEVID

Function name	**uint32_t HAL_GetDEVID (void)**
Function description	Return the device identifier.
Return values	• **Device:** identifier

HAL_GetUIDw0

Function name	**uint32_t HAL_GetUIDw0 (void)**
Function description	Return the first word of the unique device identifier (UID based on 96 bits)
Return values	• **Device:** identifier

HAL_GetUIDw1

Function name	**uint32_t HAL_GetUIDw1 (void)**
Function description	Return the second word of the unique device identifier (UID based on 96 bits)
Return values	• **Device:** identifier

HAL_GetUIDw2

Function name	**uint32_t HAL_GetUIDw2 (void)**
Function description	Return the third word of the unique device identifier (UID based on 96 bits)
Return values	• **Device:** identifier

HAL_DBGMCU_EnableDBGSleepMode

Function name	**void HAL_DBGMCU_EnableDBGSleepMode (void)**
Function description	Enable the Debug Module during SLEEP mode.
Return values	• **None:**

HAL_DBGMCU_DisableDBGSleepMode

Function name	**void HAL_DBGMCU_DisableDBGSleepMode (void)**
Function description	Disable the Debug Module during SLEEP mode.
Return values	• **None:**

HAL_DBGMCU_EnableDBGStopMode

Function name	**void HAL_DBGMCU_EnableDBGStopMode (void)**
Function description	Enable the Debug Module during STOP0/STOP1/STOP2 modes.
Return values	• **None:**

HAL_DBGMCU_DisableDBGStopMode

Function name	**void HAL_DBGMCU_DisableDBGStopMode (void)**
Function description	Disable the Debug Module during STOP0/STOP1/STOP2 modes.
Return values	• **None:**

HAL_DBGMCU_EnableDBGStandbyMode

Function name	**void HAL_DBGMCU_EnableDBGStandbyMode (void)**
Function description	Enable the Debug Module during STANDBY mode.
Return values	• **None:**

HAL_DBGMCU_DisableDBGStandbyMode

Function name **void HAL_DBGMCU_DisableDBGStandbyMode (void)**

Function description Disable the Debug Module during STANDBY mode.

Return values
- **None:**

HAL_SYSCFG_SRAM2Erase

Function name **void HAL_SYSCFG_SRAM2Erase (void)**

Function description Start a hardware SRAM2 erase operation.

Return values
- **None:**

Notes
- As long as SRAM2 is not erased the SRAM2ER bit will be set. This bit is automatically reset at the end of the SRAM2 erase operation.

HAL_SYSCFG_EnableMemorySwappingBank

Function name **void HAL_SYSCFG_EnableMemorySwappingBank (void)**

Function description Enable the Internal FLASH Bank Swapping.

Return values
- **None:**

Notes
- This function can be used only for STM32L4xx devices.
- Flash Bank2 mapped at 0x08000000 (and aliased @0x00000000) and Flash Bank1 mapped at 0x08100000 (and aliased at 0x00100000)

HAL_SYSCFG_DisableMemorySwappingBank

Function name **void HAL_SYSCFG_DisableMemorySwappingBank (void)**

Function description Disable the Internal FLASH Bank Swapping.

Return values
- **None:**

Notes
- This function can be used only for STM32L4xx devices.
- The default state: Flash Bank1 mapped at 0x08000000 (and aliased @0x0000 0000) and Flash Bank2 mapped at 0x08100000 (and aliased at 0x00100000)

HAL_SYSCFG_VREFBUF_VoltageScalingConfig

Function name **void HAL_SYSCFG_VREFBUF_VoltageScalingConfig (uint32_t VoltageScaling)**

Function description Configure the internal voltage reference buffer voltage scale.

Parameters
- **VoltageScaling:** specifies the output voltage to achieve This parameter can be one of the following values:
  - SYSCFG_VREFBUF_VOLTAGE_SCALE0: VREF_OUT1 around 2.048 V. This requires VDDA equal to or higher than 2.4 V.
  - SYSCFG_VREFBUF_VOLTAGE_SCALE1: VREF_OUT2 around 2.5 V. This requires VDDA equal to

or higher than 2.8 V.

Return values
- **None:**

HAL_SYSCFG_VREFBUF_HighImpedanceConfig

Function name **void HAL_SYSCFG_VREFBUF_HighImpedanceConfig (uint32_t Mode)**

Function description Configure the internal voltage reference buffer high impedance mode.

Parameters
- **Mode:** specifies the high impedance mode This parameter can be one of the following values:
  - SYSCFG_VREFBUF_HIGH_IMPEDANCE_DISABLE: VREF+ pin is internally connect to VREFINT output.
  - SYSCFG_VREFBUF_HIGH_IMPEDANCE_ENABLE: VREF+ pin is high impedance.

Return values
- **None:**

HAL_SYSCFG_VREFBUF_TrimmingConfig

Function name **void HAL_SYSCFG_VREFBUF_TrimmingConfig (uint32_t TrimmingValue)**

Function description Tune the Internal Voltage Reference buffer (VREFBUF).

Return values
- **None:**

HAL_SYSCFG_EnableVREFBUF

Function name **HAL_StatusTypeDef HAL_SYSCFG_EnableVREFBUF (void)**

Function description Enable the Internal Voltage Reference buffer (VREFBUF).

Return values
- **HAL_OK/HAL_TIMEOUT:**

HAL_SYSCFG_DisableVREFBUF

Function name **void HAL_SYSCFG_DisableVREFBUF (void)**

Function description Disable the Internal Voltage Reference buffer (VREFBUF).

Return values
- **None:**

HAL_SYSCFG_EnableIOAnalogSwitchBooster

Function name **void HAL_SYSCFG_EnableIOAnalogSwitchBooster (void)**

Function description Enable the I/O analog switch voltage booster.

Return values
- **None:**

HAL_SYSCFG_DisableIOAnalogSwitchBooster

Function name **void HAL_SYSCFG_DisableIOAnalogSwitchBooster (void)**

Function description Disable the I/O analog switch voltage booster.

Return values
- **None:**

5.2 HAL Firmware driver defines

5.2.1 HAL

DBGMCU Exported Macros

__HAL_DBGMCU_FREEZE_TIM2
__HAL_DBGMCU_UNFREEZE_TIM2
__HAL_DBGMCU_FREEZE_TIM3
__HAL_DBGMCU_UNFREEZE_TIM3
__HAL_DBGMCU_FREEZE_TIM4
__HAL_DBGMCU_UNFREEZE_TIM4
__HAL_DBGMCU_FREEZE_TIM5
__HAL_DBGMCU_UNFREEZE_TIM5
__HAL_DBGMCU_FREEZE_TIM6
__HAL_DBGMCU_UNFREEZE_TIM6
__HAL_DBGMCU_FREEZE_TIM7
__HAL_DBGMCU_UNFREEZE_TIM7
__HAL_DBGMCU_FREEZE_RTC
__HAL_DBGMCU_UNFREEZE_RTC
__HAL_DBGMCU_FREEZE_WWDG
__HAL_DBGMCU_UNFREEZE_WWDG
__HAL_DBGMCU_FREEZE_IWDG
__HAL_DBGMCU_UNFREEZE_IWDG
__HAL_DBGMCU_FREEZE_I2C1_TIMEOUT
__HAL_DBGMCU_UNFREEZE_I2C1_TIMEOUT
__HAL_DBGMCU_FREEZE_I2C2_TIMEOUT
__HAL_DBGMCU_UNFREEZE_I2C2_TIMEOUT
__HAL_DBGMCU_FREEZE_I2C3_TIMEOUT
__HAL_DBGMCU_UNFREEZE_I2C3_TIMEOUT
__HAL_DBGMCU_FREEZE_I2C4_TIMEOUT
__HAL_DBGMCU_UNFREEZE_I2C4_TIMEOUT
__HAL_DBGMCU_FREEZE_CAN1
__HAL_DBGMCU_UNFREEZE_CAN1
__HAL_DBGMCU_FREEZE_LPTIM1
__HAL_DBGMCU_UNFREEZE_LPTIM1
__HAL_DBGMCU_FREEZE_LPTIM2

__HAL_DBGMCU_UNFREEZE_LPTIM2

__HAL_DBGMCU_FREEZE_TIM1

__HAL_DBGMCU_UNFREEZE_TIM1

__HAL_DBGMCU_FREEZE_TIM8

__HAL_DBGMCU_UNFREEZE_TIM8

__HAL_DBGMCU_FREEZE_TIM15

__HAL_DBGMCU_UNFREEZE_TIM15

__HAL_DBGMCU_FREEZE_TIM16

__HAL_DBGMCU_UNFREEZE_TIM16

__HAL_DBGMCU_FREEZE_TIM17

__HAL_DBGMCU_UNFREEZE_TIM17

HAL state definition

HAL_SMBUS_STATE_RESET	SMBUS not yet initialized or disabled
HAL_SMBUS_STATE_READY	SMBUS initialized and ready for use
HAL_SMBUS_STATE_BUSY	SMBUS internal process is ongoing
HAL_SMBUS_STATE_MASTER_BUSY_TX	Master Data Transmission process is ongoing
HAL_SMBUS_STATE_MASTER_BUSY_RX	Master Data Reception process is ongoing
HAL_SMBUS_STATE_SLAVE_BUSY_TX	Slave Data Transmission process is ongoing
HAL_SMBUS_STATE_SLAVE_BUSY_RX	Slave Data Reception process is ongoing
HAL_SMBUS_STATE_TIMEOUT	Timeout state
HAL_SMBUS_STATE_ERROR	Reception process is ongoing
HAL_SMBUS_STATE_LISTEN	Address Listen Mode is ongoing

Boot Mode

SYSCFG_BOOT_MAINFLASH

SYSCFG_BOOT_SYSTEMFLASH

SYSCFG_BOOT_FMC

SYSCFG_BOOT_SRAM

SYSCFG_BOOT_OCTOPSPI1

SYSCFG_BOOT_OCTOPSPI2

SYSCFG Exported Macros

__HAL_SYSCFG_REMAPMEMORY_FLASH

__HAL_SYSCFG_REMAPMEMORY_SYSTEMFLASH

__HAL_SYSCFG_REMAPMEMORY_SRAM

__HAL_SYSCFG_REMAPMEMORY_FMC

__HAL_SYSCFG_REMAPMEMORY_OCTOSPI1

__HAL_SYSCFG_REMAPMEMORY_OCTOSPI2

__HAL_SYSCFG_GET_BOOT_MODE

Description:

- Return the boot mode as configured by user.

Return value:

- The: boot mode as configured by user. The returned value can be one of the following values:
  - SYSCFG_BOOT_MAINFLASH
  - SYSCFG_BOOT_SYSTEMFLASH
  - SYSCFG_BOOT_SRAM
  - SYSCFG_BOOT_QUADSPI

__HAL_SYSCFG_SRAM2_WRP_1_31_ENABLE

Description:

- SRAM2 page 0 to 31 write protection enable macro.

Parameters:

- __SRAM2WRP__: This parameter can be a combination of values of

Notes:

- Write protection can only be disabled by a system reset

__HAL_SYSCFG_SRAM2_WRP_32_63_ENABLE

Description:

- SRAM2 page 32 to 63 write protection enable macro.

Parameters:

- __SRAM2WRP__: This parameter can be a combination of values of

Notes:

- Write protection can only be disabled by a system reset

__HAL_SYSCFG_SRAM2_WRP_UNLOCK

Notes:

- Writing a wrong key reactivates the write protection

__HAL_SYSCFG_SRAM2_ERASE

Notes:

- __SYSCFG_GET_FLAG(SYSCFG_FLAG_SRAM2_BUSY) may be used to check end of erase

__HAL_SYSCFG_FPU_INTERRUPT_ENABLE

Description:

- Floating Point Unit interrupt enable/disable macros.

Parameters:

- __INTERRUPT__: This parameter can be a value of

__HAL_SYSCFG_FPU_INTERRUPT_DISABLE

__HAL_SYSCFG_BREAK_ECC_LOCK

Notes:

- The selected configuration is locked and can be unlocked only by system reset.

Enable and lock the connection of Flash ECC error connection to TIM1/8/15/16/17 Break input.

__HAL_SYSCFG_BREAK_LOCKUP_LOCK

Notes:

- The selected configuration is locked and can be unlocked only by system reset.

Enable and lock the connection of Cortex-M4 LOCKUP (Hardfault) output to TIM1/8/15/16/17 Break input.

__HAL_SYSCFG_BREAK_PVD_LOCK

Notes:

- The selected configuration is locked and can be unlocked only by system reset.

Enable and lock the PVD connection to Timer1/8/15/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR_CR2 register.

__HAL_SYSCFG_BREAK_SRAM2PARITY_LOCK

Notes:

- The selected configuration is locked and can be unlocked by system reset.

Enable and lock the SRAM2 parity error signal connection to TIM1/8/15/16/17 Break input.

__HAL_SYSCFG_GET_FLAG

Description:

- Check SYSCFG flag is set or not.

Parameters:

- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
  - SYSCFG_FLAG_SRAM2_PE SRAM2 Parity Error Flag
  - SYSCFG_FLAG_SRAM2_BUSY SRAM2 Erase Ongoing

Return value:

- The: new state of __FLAG__ (TRUE or

FALSE).

__HAL_SYSCFG_CLEAR_FLAG

__HAL_SYSCFG_FASTMODEPLUS_ENABLE

Description:

- Fast-mode Plus driving capability enable/disable macros.

Parameters:

- __FASTMODEPLUS__: This parameter can be a value of:
  - SYSCFG_FASTMODEPLUS_PB6 Fast-mode Plus driving capability activation on PB6
  - SYSCFG_FASTMODEPLUS_PB7 Fast-mode Plus driving capability activation on PB7
  - SYSCFG_FASTMODEPLUS_PB8 Fast-mode Plus driving capability activation on PB8
  - SYSCFG_FASTMODEPLUS_PB9 Fast-mode Plus driving capability activation on PB9

__HAL_SYSCFG_FASTMODEPLUS_DISABLE

Fast-mode Plus on GPIO

SYSCFG_FASTMODEPLUS_PB6	Enable Fast-mode Plus on PB6
SYSCFG_FASTMODEPLUS_PB7	Enable Fast-mode Plus on PB7
SYSCFG_FASTMODEPLUS_PB8	Enable Fast-mode Plus on PB8
SYSCFG_FASTMODEPLUS_PB9	Enable Fast-mode Plus on PB9

Flags

SYSCFG_FLAG_SRAM2_PE	SRAM2 parity error
SYSCFG_FLAG_SRAM2_BUSY	SRAM2 busy by erase operation

FPU Interrupts

SYSCFG_IT_FPU_IOC	Floating Point Unit Invalid operation Interrupt
SYSCFG_IT_FPU_DZC	Floating Point Unit Divide-by-zero Interrupt
SYSCFG_IT_FPU_UFC	Floating Point Unit Underflow Interrupt
SYSCFG_IT_FPU_OFC	Floating Point Unit Overflow Interrupt
SYSCFG_IT_FPU_IDC	Floating Point Unit Input denormal Interrupt
SYSCFG_IT_FPU_IXC	Floating Point Unit Inexact Interrupt

SRAM2 Page Write protection (0 to 31)

SYSCFG_SRAM2WRP_PAGE0	SRAM2 Write protection page 0
SYSCFG_SRAM2WRP_PAGE1	SRAM2 Write protection page 1
SYSCFG_SRAM2WRP_PAGE2	SRAM2 Write protection page 2

SYSCFG_SRAM2WRP_PAGE3	SRAM2 Write protection page 3
SYSCFG_SRAM2WRP_PAGE4	SRAM2 Write protection page 4
SYSCFG_SRAM2WRP_PAGE5	SRAM2 Write protection page 5
SYSCFG_SRAM2WRP_PAGE6	SRAM2 Write protection page 6
SYSCFG_SRAM2WRP_PAGE7	SRAM2 Write protection page 7
SYSCFG_SRAM2WRP_PAGE8	SRAM2 Write protection page 8
SYSCFG_SRAM2WRP_PAGE9	SRAM2 Write protection page 9
SYSCFG_SRAM2WRP_PAGE10	SRAM2 Write protection page 10
SYSCFG_SRAM2WRP_PAGE11	SRAM2 Write protection page 11
SYSCFG_SRAM2WRP_PAGE12	SRAM2 Write protection page 12
SYSCFG_SRAM2WRP_PAGE13	SRAM2 Write protection page 13
SYSCFG_SRAM2WRP_PAGE14	SRAM2 Write protection page 14
SYSCFG_SRAM2WRP_PAGE15	SRAM2 Write protection page 15
SYSCFG_SRAM2WRP_PAGE16	SRAM2 Write protection page 16
SYSCFG_SRAM2WRP_PAGE17	SRAM2 Write protection page 17
SYSCFG_SRAM2WRP_PAGE18	SRAM2 Write protection page 18
SYSCFG_SRAM2WRP_PAGE19	SRAM2 Write protection page 19
SYSCFG_SRAM2WRP_PAGE20	SRAM2 Write protection page 20
SYSCFG_SRAM2WRP_PAGE21	SRAM2 Write protection page 21
SYSCFG_SRAM2WRP_PAGE22	SRAM2 Write protection page 22
SYSCFG_SRAM2WRP_PAGE23	SRAM2 Write protection page 23
SYSCFG_SRAM2WRP_PAGE24	SRAM2 Write protection page 24
SYSCFG_SRAM2WRP_PAGE25	SRAM2 Write protection page 25
SYSCFG_SRAM2WRP_PAGE26	SRAM2 Write protection page 26
SYSCFG_SRAM2WRP_PAGE27	SRAM2 Write protection page 27
SYSCFG_SRAM2WRP_PAGE28	SRAM2 Write protection page 28
SYSCFG_SRAM2WRP_PAGE29	SRAM2 Write protection page 29
SYSCFG_SRAM2WRP_PAGE30	SRAM2 Write protection page 30
SYSCFG_SRAM2WRP_PAGE31	SRAM2 Write protection page 31

SRAM2 Page Write protection (32 to 63)

SYSCFG_SRAM2WRP_PAGE32	SRAM2 Write protection page 32
SYSCFG_SRAM2WRP_PAGE33	SRAM2 Write protection page 33
SYSCFG_SRAM2WRP_PAGE34	SRAM2 Write protection page 34
SYSCFG_SRAM2WRP_PAGE35	SRAM2 Write protection page 35
SYSCFG_SRAM2WRP_PAGE36	SRAM2 Write protection page 36
SYSCFG_SRAM2WRP_PAGE37	SRAM2 Write protection page 37

SYSCFG_SRAM2WRP_PAGE38	SRAM2 Write protection page 38
SYSCFG_SRAM2WRP_PAGE39	SRAM2 Write protection page 39
SYSCFG_SRAM2WRP_PAGE40	SRAM2 Write protection page 40
SYSCFG_SRAM2WRP_PAGE41	SRAM2 Write protection page 41
SYSCFG_SRAM2WRP_PAGE42	SRAM2 Write protection page 42
SYSCFG_SRAM2WRP_PAGE43	SRAM2 Write protection page 43
SYSCFG_SRAM2WRP_PAGE44	SRAM2 Write protection page 44
SYSCFG_SRAM2WRP_PAGE45	SRAM2 Write protection page 45
SYSCFG_SRAM2WRP_PAGE46	SRAM2 Write protection page 46
SYSCFG_SRAM2WRP_PAGE47	SRAM2 Write protection page 47
SYSCFG_SRAM2WRP_PAGE48	SRAM2 Write protection page 48
SYSCFG_SRAM2WRP_PAGE49	SRAM2 Write protection page 49
SYSCFG_SRAM2WRP_PAGE50	SRAM2 Write protection page 50
SYSCFG_SRAM2WRP_PAGE51	SRAM2 Write protection page 51
SYSCFG_SRAM2WRP_PAGE52	SRAM2 Write protection page 52
SYSCFG_SRAM2WRP_PAGE53	SRAM2 Write protection page 53
SYSCFG_SRAM2WRP_PAGE54	SRAM2 Write protection page 54
SYSCFG_SRAM2WRP_PAGE55	SRAM2 Write protection page 55
SYSCFG_SRAM2WRP_PAGE56	SRAM2 Write protection page 56
SYSCFG_SRAM2WRP_PAGE57	SRAM2 Write protection page 57
SYSCFG_SRAM2WRP_PAGE58	SRAM2 Write protection page 58
SYSCFG_SRAM2WRP_PAGE59	SRAM2 Write protection page 59
SYSCFG_SRAM2WRP_PAGE60	SRAM2 Write protection page 60
SYSCFG_SRAM2WRP_PAGE61	SRAM2 Write protection page 61
SYSCFG_SRAM2WRP_PAGE62	SRAM2 Write protection page 62
SYSCFG_SRAM2WRP_PAGE63	SRAM2 Write protection page 63

VREFBUF High Impedance

SYSCFG_VREFBUF_HIGH_IMPEDANCE_DISABLE	VREF_plus pin is internally connected to Voltage reference buffer output
SYSCFG_VREFBUF_HIGH_IMPEDANCE_ENABLE	VREF_plus pin is high impedance

VREFBUF Voltage Scale

SYSCFG_VREFBUF_VOLTAGE_SCALE0	Voltage reference scale 0 (VREF_OUT1)
SYSCFG_VREFBUF_VOLTAGE_SCALE1	Voltage reference scale 1 (VREF_OUT2)

6 HAL ADC Generic Driver

6.1 ADC Firmware driver registers structures

6.1.1 ADC_OversamplingTypeDef

Data Fields

- ***uint32_t Ratio***
- ***uint32_t RightBitShift***
- ***uint32_t TriggeredMode***
- ***uint32_t OversamplingStopReset***

Field Documentation

- ***uint32_t ADC_OversamplingTypeDef::Ratio***
  Configures the oversampling ratio. This parameter can be a value of ***ADC_Oversampling_Ratio***
- ***uint32_t ADC_OversamplingTypeDef::RightBitShift***
  Configures the division coefficient for the Oversampler. This parameter can be a value of ***ADC_Right_Bit_Shift***
- ***uint32_t ADC_OversamplingTypeDef::TriggeredMode***
  Selects the regular triggered oversampling mode. This parameter can be a value of ***ADC_Triggered_Oversampling_Mode***
- ***uint32_t ADC_OversamplingTypeDef::OversamplingStopReset***
  Selects the regular oversampling mode. The oversampling is either temporary stopped or reset upon an injected sequence interruption. If oversampling is enabled on both regular and injected groups, this parameter is discarded and forced to setting "ADC_REGOVERSAMPLING_RESUMED_MODE" (the oversampling buffer is zeroed during injection sequence). This parameter can be a value of ***ADC_Regular_Oversampling_Mode***

6.1.2 ADC_InitTypeDef

Data Fields

- ***uint32_t ClockPrescaler***
- ***uint32_t Resolution***
- ***uint32_t DataAlign***
- ***uint32_t ScanConvMode***
- ***uint32_t EOCSelection***
- ***uint32_t LowPowerAutoWait***
- ***uint32_t ContinuousConvMode***
- ***uint32_t NbrOfConversion***
- ***uint32_t DiscontinuousConvMode***
- ***uint32_t NbrOfDiscConversion***
- ***uint32_t ExternalTrigConv***
- ***uint32_t ExternalTrigConvEdge***
- ***uint32_t DMAContinuousRequests***
- ***uint32_t Overrun***
- ***uint32_t OversamplingMode***
- ***ADC_OversamplingTypeDef Oversampling***
- ***uint32_t DFSDMConfig***