Description of STM32F4 HAL and low-layer drivers

Introduction

STM32Cube is an STMicroelectronics original initiative to significantly improve developer productivity by reducing development effort, time and cost. STM32Cube covers the STM32 portfolio.

STM32Cube includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per Series (such as STM32CubeF4 for STM32F4 Series)
  - The STM32Cube HAL, STM32 abstraction layer embedded software ensuring maximized portability across the STM32 portfolio. HAL APIs are available for all peripherals.
  - Low-layer APIs (LL) offering a fast light-weight expert-oriented layer which is closer to the hardware than the HAL. LL APIs are available only for a set of peripherals.
  - A consistent set of middleware components such as RTOS, USB, TCP/IP and Graphics.
  - All embedded software utilities, delivered with a full set of examples.

The HAL driver layer provides a simple, generic multi-instance set of APIs (application programming interfaces) to interact with the upper layer (application, libraries and stacks). The HAL driver APIs are split into two categories: generic APIs, which provide common and generic functions for all the STM32 series and extension APIs, which include specific and customized functions for a given line or part number. The HAL drivers include a complete set of ready-to-use APIs that simplify the user application implementation. For example, the communication peripherals contain APIs to initialize and configure the peripheral, manage data transfers in polling mode, handle interrupts or DMA, and manage communication errors. The HAL drivers are feature-oriented instead of IP-oriented. For example, the timer APIs are split into several categories following the IP functions, such as basic timer, capture and pulse width modulation (PWM). The HAL driver layer implements run-time failure detection by checking the input values of all functions. Such dynamic checking enhances the firmware robustness. Run-time detection is also suitable for user application development and debugging.

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

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

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

The HAL- and LL-driver source code is developed in Strict ANSI-C, which makes it independent of the development tools. It is checked with the CodeSonar® static analysis tool. It is fully documented.

It is compliant with MISRA C®:2004 standard.

This user manual is structured as follows:

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

1 General information

The STM32CubeF4 MCU Package runs on STM32F4 32-bit microcontrollers based on the Arm® Cortex®-M processor.

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

2 Acronyms and definitions

Table 1. Acronyms and definitions

Acronym	Definition
ADC	Analog-to-digital converter
AES	Advanced encryption standard
ANSI	American national standards institute
API	Application programming interface
BSP	Board support package
CAN	Controller area network
CEC	Consumer electronic controller
CMSIS	Cortex microcontroller software interface standard
COMP	Comparator
CORDIC	Trigonometric calculation unit
CPU	Central processing unit
CRC	CRC calculation unit
CRYP	Cryptographic processor
CSS	Clock security system
DAC	Digital to analog converter
DLYB	Delay block
DCMI	Digital camera interface
DFSDM	Digital filter sigma delta modulator
DMA	Direct memory access
DMAMUX	Direct memory access request multiplexer
DSI	Display serial interface
DTS	Digital temperature sensor
ESE	Security enable Flash user option bit
ETH	Ethernet controller
EXTI	External interrupt/event controller
FDCAN	Flexible data-rate controller area network unit
FLASH	Flash memory
FMAC	Filtering mathematical calculation unit
FMC	Flexible memory controller
FW	Firewall
GFXMMU	Chrom-GRC
GPIO	General purpose I/Os
GTZC	Global security controller
GTZC-MPCBB	GTZC block-based memory protection controller
GTZC-MPCWM	GTZC watermark memory protection controller
GTZC-TZIC	Security illegal access controller
GTZC-TZSC	Security access controller

Acronym	Definition
HAL	Hardware abstraction layer
HASH	Hash processor
HCD	USB host controller driver
HRTIM	High-resolution timer
I2C	Inter-integrated circuit
I2S	Inter-integrated sound
ICACHE	Instruction cache
IRDA	Infrared data association
IWDG	Independent watchdog
JPEG	Joint photographic experts group
LCD	Liquid crystal display controler
LTDC	LCD TFT display controller
LPTIM	Low-power timer
LPUART	Low-power universal asynchronous receiver/transmitter
MCO	Microcontroller clock output
MDIOS	Management data input/output (MDIO) slave
MDMA	Master direct memory access
MMC	MultiMediaCard
MPU	Memory protection unit
MSP	MCU specific package
NAND	NAND Flash memory
NOR	NOR Flash memory
NVIC	Nested vectored interrupt controller
OCTOSPI	Octo-SPI interface
OPAMP	Operational amplifier
OTFDEC	On-the-fly decryption engine
OTG-FS	USB on-the-go full-speed
PKA	Public key accelerator
PCD	USB peripheral controller driver
PPP	STM32 peripheral or block
PSSI	Parallel synchronous slave interface
PWR	Power controller
QSPI	Quad-SPI Flash memory
RAMECC	RAM ECC monitoring
RCC	Reset and clock controller
RNG	Random number generator
RTC	Real-time clock
SAI	Serial audio interface
SD	Secure digital
SDMMC	SD/SDIO/MultiMediaCard card host interface
SMARTCARD	Smartcard IC

Acronym	Definition
SMBUS	System management bus
SPI	Serial peripheral interface
SPDIFRX	SPDIF-RX Receiver interface
SRAM	SRAM external memory
SWPMI	Serial wire protocol master interface
SysTick	System tick timer
TIM	Advanced-control, general-purpose or basic timer
TSC	Touch sensing controller
UART	Universal asynchronous receiver/transmitter
UCPD	USB Type-C® and Power Delivery interface
USART	Universal synchronous receiver/transmitter
VREFBUF	Voltage reference buffer
WWDG	Window watchdog
USB	Universal serial bus

4 Overview of low-layer drivers

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

The LL drivers feature:

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

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

4.1 Low-layer files

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

Table 16. LL driver files

File	Description
stm32f4xx_ll_bus.h	This is the h-source file for core bus control and peripheral clock activation and deactivation *Example: LL_AHB2_GRP1_EnableClock*
stm32f4xx_ll_ppp.h/.c	stm32f4xx_ll_ppp.c provides peripheral initialization functions such as LL_PPP_Init(), LL_PPP_StructInit(), LL_PPP_DeInit(). All the other APIs are definined within stm32f4xx_ll_ppp.h file. The low-layer PPP driver is a standalone module. To use it, the application must include it in the stm32f4xx_ll_ppp.h file.
stm32f4xx_ll_cortex.h	Cortex-M related register operation APIs including the Systick, Low power (such as LL_SYSTICK_xxxxx and LL_LPM_xxxxx "Low Power Mode")
stm32f4xx_ll_utils.h/.c	This file covers the generic APIs: • Read of device unique ID and electronic signature • Timebase and delay management • System clock configuration.
stm32f4xx_ll_system.h	System related operations. Example: LL_SYSCFG_xxx, LL_DBGMCU_xxx and LL_FLASH_xxx and LL_VREFBUF_xxx
stm32_assert_template.h	Template file allowing to define the assert_param macro, that is used when run-time checking is enabled. This file is required only when the LL drivers are used in standalone mode (without calling the HAL APIs). It should be copied to the application folder and renamed to stm32_assert.h.

Note: There is no configuration file for the LL drivers.

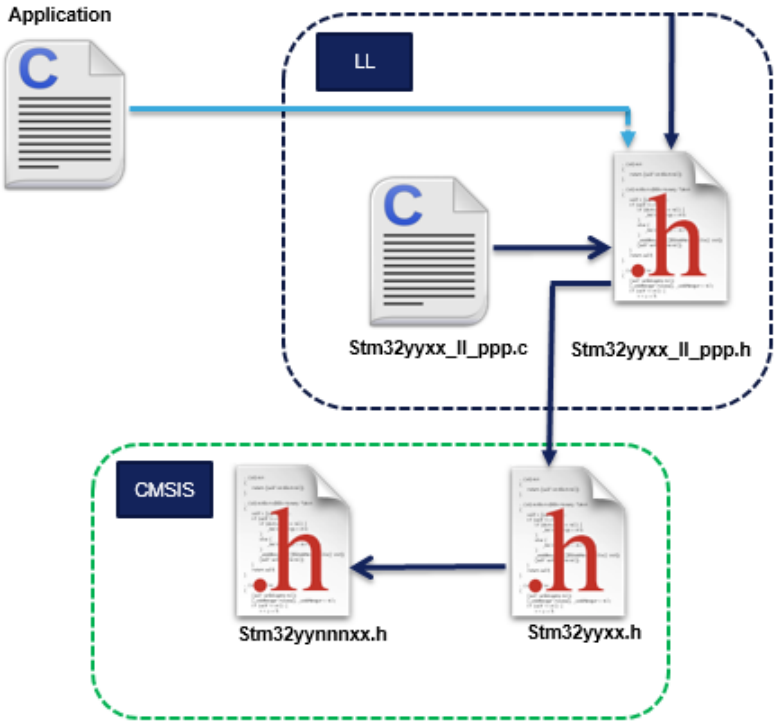
The low-layer files are located in the same HAL driver folder.

Figure 8. Low-layer driver folders

In general, low-layer drivers include only the STM32 CMSIS device file.

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

Figure 9. Low-layer driver CMSIS files



Application files have to include only the used low-layer driver header files.

4.2 Overview of low-layer APIs and naming rules

4.2.1 Peripheral initialization functions

The LL drivers offer three sets of initialization functions. They are defined in stm32f4xx_ll_ppp.c file:

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

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

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

Table 17. Common peripheral initialization functions

Functions	Return Type	Parameters	Description
LL_PPP_Init	*ErrorStatus*	• *PPP_TypeDef* PPPx* • *LL_PPP_InitTypeDef* PPP_InitStruct*	Initializes the peripheral main features according to the parameters specified in PPP_InitStruct. Example: LL_USART_Init(USART_TypeDef *USARTx, LL_USART_InitTypeDef *USART_InitStruct)
LL_PPP_StructInit	*void*	• *LL_PPP_InitTypeDef* PPP_InitStruct*	Fills each PPP_InitStruct member with its default value. Example. LL_USART_StructInit(LL_USART_InitTypeDef *USART_InitStruct)
LL_PPP_DeInit	*ErrorStatus*	• *PPP_TypeDef* PPPx*	De-initializes the peripheral registers, that is restore them to their default reset values. Example. LL_USART_DeInit(USART_TypeDef *USARTx)

Additional functions are available for some peripherals (refer to Table 18. Optional peripheral initialization functions).

Table 18. Optional peripheral initialization functions

Functions	Return Type	Parameters	Examples
LL_PPP{_CATEGORY}_Init	*ErrorStatus*	• *PPP_TypeDef* PPPx* • *LL_PPP{_CATEGORY}_InitTypeDef* PPP{_CATEGORY}_InitStruct*	Initializes peripheral features according to the parameters specified in PPP_InitStruct. Example: LL_ADC_INJ_Init(ADC_TypeDef *ADCx, LL_ADC_INJ_InitTypeDef *ADC_INJ_InitStruct) LL_RTC_TIME_Init(RTC_TypeDef *RTCx, uint32_t RTC_Format, LL_RTC_TimeTypeDef *RTC_TimeStruct) LL_RTC_DATE_Init(RTC_TypeDef *RTCx, uint32_t RTC_Format, LL_RTC_DateTypeDef *RTC_DateStruct) LL_TIM_IC_Init(TIM_TypeDef* TIMx, uint32_t Channel, LL_TIM_IC_InitTypeDef* TIM_IC_InitStruct) LL_TIM_ENCODER_Init(TIM_TypeDef* TIMx, LL_TIM_ENCODER_InitTypeDef* TIM_EncoderInitStruct)
LL_PPP{_CATEGORY}_StructInit	*void*	*LL_PPP{_CATEGORY}_InitTypeDef* PPP{_CATEGORY}_InitStruct*	Fills each *PPP{_CATEGORY}_InitStruct* member with its default value. Example: LL_ADC_INJ_StructInit(LL_ADC_INJ_InitTypeDef *ADC_INJ_InitStruct)
LL_PPP_CommonInit	*ErrorStatus*	• *PPP_TypeDef* PPPx* • *LL_PPP_CommonInitTypeDef* PPP_CommonInitStruct*	Initializes the common features shared between different instances of the same peripheral. Example: LL_ADC_CommonInit(ADC_Common_TypeDef *ADCxy_COMMON, LL_ADC_CommonInitTypeDef *ADC_CommonInitStruct)
LL_PPP_CommonStructInit	*void*	*LL_PPP_CommonInitTypeDef* PPP_CommonInitStruct*	Fills each *PPP_CommonInitStruct* member with its default value Example: LL_ADC_CommonStructInit(LL_ADC_CommonInitTypeDef *ADC_CommonInitStruct)
LL_PPP_ClockInit	*ErrorStatus*	• *PPP_TypeDef* PPPx* • *LL_PPP_ClockInitTypeDef* PPP_ClockInitStruct*	Initializes the peripheral clock configuration in synchronous mode. Example: LL_USART_ClockInit(USART_TypeDef *USARTx, LL_USART_ClockInitTypeDef *USART_ClockInitStruct)

Functions	Return Type	Parameters	Examples
LL_PPP_ClockStructInit	*void*	*LL_PPP_ClockInitTypeDef* PPP_ClockInitStruct*	Fills each *PPP_ClockInitStruct* member with its default value Example: LL_USART_ClockStructInit(LL_USART_ClockInitTypeDef *USART_ClockInitStruct)

4.2.1.1 *Run-time checking*

Like HAL drivers, LL initialization functions implement run-time failure detection by checking the input values of all LL driver functions. For more details please refer to Section 3.12.4.3 Run-time checking.

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

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

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

4.2.2 Peripheral register-level configuration functions

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

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

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

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

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

Name	Examples
LL_PPP_{_CATEGORY}_ActionItem_BITNAME *LL_PPP{_CATEGORY}_IsItem_BITNAME_Action*	• LL_RCC_IsActiveFlag_LSIRDY • LL_RCC_IsActiveFlag_FWRST() • LL_ADC_ClearFlag_EOC(ADC1) • LL_DMA_ClearFlag_TCx(DMA_TypeDef* DMAx)

Table 20. Available function formats

Item	Action	Format
Flag	Get	*LL_PPP_IsActiveFlag_BITNAME*
	Clear	*LL_PPP_ClearFlag_BITNAME*
Interrupts	Enable	*LL_PPP_EnableIT_BITNAME*
	Disable	*LL_PPP_DisableIT_BITNAME*
	Get	*LL_PPP_IsEnabledIT_BITNAME*
DMA	Enable	*LL_PPP_EnableDMAReq_BITNAME*
	Disable	*LL_PPP_DisableDMAReq_BITNAME*
	Get	*LL_PPP_IsEnabledDMAReq_BITNAME*

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

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

Table 21. Peripheral clock activation/deactivation management

Name	Examples
LL_BUS_GRPx_ActionClock{Mode}	• *LL_AHB2_GRP1_EnableClock (LL_AHB2_GRP1_PERIPH_GPIOA| LL_AHB2_GRP1_PERIPH_GPIOB)* • *LL_APB1_GRP1_EnableClockSleep (LL_APB1_GRP1_PERIPH_DAC1)*

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

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

Table 22. Peripheral activation/deactivation management

Name	Examples
LL_PPP{_CATEGORY}_Action{Item} *LL_PPP{_CATEGORY}_IsItemAction*	• *LL_ADC_Enable ()* • *LL_ADC_StartCalibration();* • *LL_ADC_IsCalibrationOnGoing;* • *LL_RCC_HSI_Enable ()* • *LL_RCC_HSI_IsReady()*

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

Table 23. Peripheral configuration management

Name	Examples
LL_PPP{_CATEGORY}_Set{ or Get}ConfigItem	*LL_USART_SetBaudRate (USART2, Clock, LL_USART_BAUDRATE_9600)*

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

Table 24. Peripheral register management

Name
LL_PPP_WriteReg(__INSTANCE__, __REG__, __VALUE__)
LL_PPP_ReadReg(__INSTANCE__, __REG__)
*LL_PPP_DMA_GetRegAddr (PPP_TypeDef *PPPx,{Sub Instance if any ex: Channel} , {uint32_t Propriety})*

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

73 LL ADC Generic Driver

73.1 ADC Firmware driver registers structures

73.1.1 LL_ADC_CommonInitTypeDef

LL_ADC_CommonInitTypeDef is defined in the stm32f4xx_ll_adc.h

Data Fields

- ***uint32_t CommonClock***
- ***uint32_t Multimode***
- ***uint32_t MultiDMATransfer***
- ***uint32_t MultiTwoSamplingDelay***

Field Documentation

- ***uint32_t LL_ADC_CommonInitTypeDef::CommonClock***

  Set parameter common to several ADC: Clock source and prescaler. This parameter can be a value of ***ADC_LL_EC_COMMON_CLOCK_SOURCE***This feature can be modified afterwards using unitary function **LL_ADC_SetCommonClock()**.
- ***uint32_t LL_ADC_CommonInitTypeDef::Multimode***

  Set ADC multimode configuration to operate in independent mode or multimode (for devices with several ADC instances). This parameter can be a value of ***ADC_LL_EC_MULTI_MODE***This feature can be modified afterwards using unitary function **LL_ADC_SetMultimode()**.
- ***uint32_t LL_ADC_CommonInitTypeDef::MultiDMATransfer***

  Set ADC multimode conversion data transfer: no transfer or transfer by DMA. This parameter can be a value of ***ADC_LL_EC_MULTI_DMA_TRANSFER***This feature can be modified afterwards using unitary function **LL_ADC_SetMultiDMATransfer()**.
- ***uint32_t LL_ADC_CommonInitTypeDef::MultiTwoSamplingDelay***

  Set ADC multimode delay between 2 sampling phases. This parameter can be a value of ***ADC_LL_EC_MULTI_TWOSMP_DELAY***This feature can be modified afterwards using unitary function **LL_ADC_SetMultiTwoSamplingDelay()**.

73.1.2 LL_ADC_InitTypeDef

LL_ADC_InitTypeDef is defined in the stm32f4xx_ll_adc.h

Data Fields

- ***uint32_t Resolution***
- ***uint32_t DataAlignment***
- ***uint32_t SequencersScanMode***

Field Documentation

- ***uint32_t LL_ADC_InitTypeDef::Resolution***

  Set ADC resolution. This parameter can be a value of ***ADC_LL_EC_RESOLUTION***This feature can be modified afterwards using unitary function **LL_ADC_SetResolution()**.
- ***uint32_t LL_ADC_InitTypeDef::DataAlignment***

  Set ADC conversion data alignment. This parameter can be a value of ***ADC_LL_EC_DATA_ALIGN***This feature can be modified afterwards using unitary function **LL_ADC_SetDataAlignment()**.
- ***uint32_t LL_ADC_InitTypeDef::SequencersScanMode***

  Set ADC scan selection. This parameter can be a value of ***ADC_LL_EC_SCAN_SELECTION***This feature can be modified afterwards using unitary function **LL_ADC_SetSequencersScanMode()**.

73.1.3 LL_ADC_REG_InitTypeDef

LL_ADC_REG_InitTypeDef is defined in the stm32f4xx_ll_adc.h

Data Fields

- ***uint32_t TriggerSource***
- ***uint32_t SequencerLength***

- *uint32_t SequencerDiscont*
- *uint32_t ContinuousMode*
- *uint32_t DMATransfer*

Field Documentation

- ***uint32_t LL_ADC_REG_InitTypeDef::TriggerSource***

  Set ADC group regular conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line). This parameter can be a value of ***ADC_LL_EC_REG_TRIGGER_SOURCE***

  Note:

  – On this STM32 serie, setting of external trigger edge is performed using function **LL_ADC_REG_StartConversionExtTrig()**.

  This feature can be modified afterwards using unitary function **LL_ADC_REG_SetTriggerSource()**.

- ***uint32_t LL_ADC_REG_InitTypeDef::SequencerLength***

  Set ADC group regular sequencer length. This parameter can be a value of ***ADC_LL_EC_REG_SEQ_SCAN_LENGTH***

  Note:

  – This parameter is discarded if scan mode is disabled (refer to parameter 'ADC_SequencersScanMode').

  This feature can be modified afterwards using unitary function **LL_ADC_REG_SetSequencerLength()**.

- ***uint32_t LL_ADC_REG_InitTypeDef::SequencerDiscont***

  Set ADC group regular sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks. This parameter can be a value of ***ADC_LL_EC_REG_SEQ_DISCONT_MODE***

  Note:

  – This parameter has an effect only if group regular sequencer is enabled (scan length of 2 ranks or more).

  This feature can be modified afterwards using unitary function **LL_ADC_REG_SetSequencerDiscont()**.

- ***uint32_t LL_ADC_REG_InitTypeDef::ContinuousMode***

  Set ADC continuous conversion mode on ADC group regular, whether ADC conversions are performed in single mode (one conversion per trigger) or in continuous mode (after the first trigger, following conversions launched successively automatically). This parameter can be a value of ***ADC_LL_EC_REG_CONTINUOUS_MODE*** Note: It is not possible to enable both ADC group regular continuous mode and discontinuous mode.This feature can be modified afterwards using unitary function **LL_ADC_REG_SetContinuousMode()**.

- ***uint32_t LL_ADC_REG_InitTypeDef::DMATransfer***

  Set ADC group regular conversion data transfer: no transfer or transfer by DMA, and DMA requests mode. This parameter can be a value of ***ADC_LL_EC_REG_DMA_TRANSFER***This feature can be modified afterwards using unitary function **LL_ADC_REG_SetDMATransfer()**.

73.1.4 LL_ADC_INJ_InitTypeDef

LL_ADC_INJ_InitTypeDef is defined in the stm32f4xx_ll_adc.h

Data Fields

- *uint32_t TriggerSource*
- *uint32_t SequencerLength*
- *uint32_t SequencerDiscont*
- *uint32_t TrigAuto*

Field Documentation

- ***uint32_t LL_ADC_INJ_InitTypeDef::TriggerSource***

  Set ADC group injected conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line). This parameter can be a value of ***ADC_LL_EC_INJ_TRIGGER_SOURCE***

  Note:

  – On this STM32 serie, setting of external trigger edge is performed using function **LL_ADC_INJ_StartConversionExtTrig()**.

  This feature can be modified afterwards using unitary function **LL_ADC_INJ_SetTriggerSource()**.

- ***uint32_t LL_ADC_INJ_InitTypeDef::SequencerLength***

  Set ADC group injected sequencer length. This parameter can be a value of ***ADC_LL_EC_INJ_SEQ_SCAN_LENGTH***

  Note:

  – This parameter is discarded if scan mode is disabled (refer to parameter 'ADC_SequencersScanMode').

  This feature can be modified afterwards using unitary function **LL_ADC_INJ_SetSequencerLength()**.

- ***uint32_t LL_ADC_INJ_InitTypeDef::SequencerDiscont***

  Set ADC group injected sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks. This parameter can be a value of ***ADC_LL_EC_INJ_SEQ_DISCONT_MODE***

  Note:

  – This parameter has an effect only if group injected sequencer is enabled (scan length of 2 ranks or more).

  This feature can be modified afterwards using unitary function **LL_ADC_INJ_SetSequencerDiscont()**.

- ***uint32_t LL_ADC_INJ_InitTypeDef::TrigAuto***

  Set ADC group injected conversion trigger: independent or from ADC group regular. This parameter can be a value of ***ADC_LL_EC_INJ_TRIG_AUTO*** Note: This parameter must be set to set to independent trigger if injected trigger source is set to an external trigger.This feature can be modified afterwards using unitary function **LL_ADC_INJ_SetTrigAuto()**.

73.2 ADC Firmware driver API description

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

73.2.1 Detailed description of functions

LL_ADC_DMA_GetRegAddr

Function name

__STATIC_INLINE uint32_t LL_ADC_DMA_GetRegAddr (ADC_TypeDef * ADCx, uint32_t Register)

Function description

Function to help to configure DMA transfer from ADC: retrieve the ADC register address from ADC instance and a list of ADC registers intended to be used (most commonly) with DMA transfer.

Parameters

- **ADCx:** ADC instance
- **Register:** This parameter can be one of the following values:
  – LL_ADC_DMA_REG_REGULAR_DATA
  – LL_ADC_DMA_REG_REGULAR_DATA_MULTI (1)

  (1) Available on devices with several ADC instances.

Return values

- **ADC:** register address

Notes

- These ADC registers are data registers: when ADC conversion data is available in ADC data registers, ADC generates a DMA transfer request.
- This macro is intended to be used with LL DMA driver, refer to function "LL_DMA_ConfigAddresses()". Example: LL_DMA_ConfigAddresses(DMA1, LL_DMA_CHANNEL_1, LL_ADC_DMA_GetRegAddr(ADC1, LL_ADC_DMA_REG_REGULAR_DATA), (uint32_t)&< array or variable >, LL_DMA_DIRECTION_PERIPH_TO_MEMORY);
- For devices with several ADC: in multimode, some devices use a different data register outside of ADC instance scope (common data register). This macro manages this register difference, only ADC instance has to be set as parameter.

Reference Manual to LL API cross reference:

- DR RDATA LL_ADC_DMA_GetRegAddr
- CDR RDATA_MST LL_ADC_DMA_GetRegAddr
- CDR RDATA_SLV LL_ADC_DMA_GetRegAddr

LL_ADC_SetCommonClock

Function name

__STATIC_INLINE void LL_ADC_SetCommonClock (ADC_Common_TypeDef * ADCxy_COMMON, uint32_t CommonClock)

Function description

Set parameter common to several ADC: Clock source and prescaler.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())
- **CommonClock:** This parameter can be one of the following values:
  - LL_ADC_CLOCK_SYNC_PCLK_DIV2
  - LL_ADC_CLOCK_SYNC_PCLK_DIV4
  - LL_ADC_CLOCK_SYNC_PCLK_DIV6
  - LL_ADC_CLOCK_SYNC_PCLK_DIV8

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCR ADCPRE LL_ADC_SetCommonClock

LL_ADC_GetCommonClock

Function name

__STATIC_INLINE uint32_t LL_ADC_GetCommonClock (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get parameter common to several ADC: Clock source and prescaler.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_CLOCK_SYNC_PCLK_DIV2
  - LL_ADC_CLOCK_SYNC_PCLK_DIV4
  - LL_ADC_CLOCK_SYNC_PCLK_DIV6
  - LL_ADC_CLOCK_SYNC_PCLK_DIV8

Reference Manual to LL API cross reference:

- CCR ADCPRE LL_ADC_GetCommonClock

LL_ADC_SetCommonPathInternalCh

Function name

__STATIC_INLINE void LL_ADC_SetCommonPathInternalCh (ADC_Common_TypeDef * ADCxy_COMMON, uint32_t PathInternal)

Function description

Set parameter common to several ADC: measurement path to internal channels (VrefInt, temperature sensor, ...).

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())
- **PathInternal:** This parameter can be a combination of the following values:
  - LL_ADC_PATH_INTERNAL_NONE
  - LL_ADC_PATH_INTERNAL_VREFINT
  - LL_ADC_PATH_INTERNAL_TEMPSENSOR
  - LL_ADC_PATH_INTERNAL_VBAT

Return values

- **None:**

Notes

- One or several values can be selected. Example: (LL_ADC_PATH_INTERNAL_VREFINT | LL_ADC_PATH_INTERNAL_TEMPSENSOR)
- Stabilization time of measurement path to internal channel: After enabling internal paths, before starting ADC conversion, a delay is required for internal voltage reference and temperature sensor stabilization time. Refer to device datasheet. Refer to literal LL_ADC_DELAY_VREFINT_STAB_US. Refer to literal LL_ADC_DELAY_TEMPSENSOR_STAB_US.
- ADC internal channel sampling time constraint: For ADC conversion of internal channels, a sampling time minimum value is required. Refer to device datasheet.

Reference Manual to LL API cross reference:

- CCR TSVREFE LL_ADC_SetCommonPathInternalCh
- CCR VBATE LL_ADC_SetCommonPathInternalCh

LL_ADC_GetCommonPathInternalCh

Function name

__STATIC_INLINE uint32_t LL_ADC_GetCommonPathInternalCh (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get parameter common to several ADC: measurement path to internal channels (VrefInt, temperature sensor, ...).

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **Returned:** value can be a combination of the following values:
  - LL_ADC_PATH_INTERNAL_NONE
  - LL_ADC_PATH_INTERNAL_VREFINT
  - LL_ADC_PATH_INTERNAL_TEMPSENSOR
  - LL_ADC_PATH_INTERNAL_VBAT

Notes

- One or several values can be selected. Example: (LL_ADC_PATH_INTERNAL_VREFINT | LL_ADC_PATH_INTERNAL_TEMPSENSOR)

Reference Manual to LL API cross reference:

- CCR TSVREFE LL_ADC_GetCommonPathInternalCh
- CCR VBATE LL_ADC_GetCommonPathInternalCh

LL_ADC_SetResolution

Function name

__STATIC_INLINE void LL_ADC_SetResolution (ADC_TypeDef * ADCx, uint32_t Resolution)

Function description

Set ADC resolution.

Parameters

- **ADCx:** ADC instance
- **Resolution:** This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RES LL_ADC_SetResolution

LL_ADC_GetResolution

Function name

__STATIC_INLINE uint32_t LL_ADC_GetResolution (ADC_TypeDef * ADCx)

Function description

Get ADC resolution.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B

Reference Manual to LL API cross reference:

- CR1 RES LL_ADC_GetResolution

LL_ADC_SetDataAlignment

Function name

__STATIC_INLINE void LL_ADC_SetDataAlignment (ADC_TypeDef * ADCx, uint32_t DataAlignment)

Function description

Set ADC conversion data alignment.

Parameters

- **ADCx:** ADC instance
- **DataAlignment:** This parameter can be one of the following values:
  - LL_ADC_DATA_ALIGN_RIGHT
  - LL_ADC_DATA_ALIGN_LEFT

Return values

- **None:**

Notes

- Refer to reference manual for alignments formats dependencies to ADC resolutions.

Reference Manual to LL API cross reference:

- CR2 ALIGN LL_ADC_SetDataAlignment

LL_ADC_GetDataAlignment

Function name

__STATIC_INLINE uint32_t LL_ADC_GetDataAlignment (ADC_TypeDef * ADCx)

Function description

Get ADC conversion data alignment.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_DATA_ALIGN_RIGHT
  - LL_ADC_DATA_ALIGN_LEFT

Notes

- Refer to reference manual for alignments formats dependencies to ADC resolutions.

Reference Manual to LL API cross reference:

- CR2 ALIGN LL_ADC_SetDataAlignment

LL_ADC_SetSequencersScanMode

Function name

__STATIC_INLINE void LL_ADC_SetSequencersScanMode (ADC_TypeDef * ADCx, uint32_t ScanMode)

Function description

Set ADC sequencers scan mode, for all ADC groups (group regular, group injected).

Parameters

- **ADCx:** ADC instance
- **ScanMode:** This parameter can be one of the following values:
  - LL_ADC_SEQ_SCAN_DISABLE
  - LL_ADC_SEQ_SCAN_ENABLE

Return values

- **None:**

Notes

- According to sequencers scan mode : If disabled: ADC conversion is performed in unitary conversion mode (one channel converted, that defined in rank 1). Configuration of sequencers of all ADC groups (sequencer scan length, ...) is discarded: equivalent to scan length of 1 rank.If enabled: ADC conversions are performed in sequence conversions mode, according to configuration of sequencers of each ADC group (sequencer scan length, ...). Refer to function LL_ADC_REG_SetSequencerLength() and to function LL_ADC_INJ_SetSequencerLength().

Reference Manual to LL API cross reference:

- CR1 SCAN LL_ADC_SetSequencersScanMode

LL_ADC_GetSequencersScanMode

Function name

__STATIC_INLINE uint32_t LL_ADC_GetSequencersScanMode (ADC_TypeDef * ADCx)

Function description

Get ADC sequencers scan mode, for all ADC groups (group regular, group injected).

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_SEQ_SCAN_DISABLE
  - LL_ADC_SEQ_SCAN_ENABLE

Notes

- According to sequencers scan mode : If disabled: ADC conversion is performed in unitary conversion mode (one channel converted, that defined in rank 1). Configuration of sequencers of all ADC groups (sequencer scan length, ...) is discarded: equivalent to scan length of 1 rank.If enabled: ADC conversions are performed in sequence conversions mode, according to configuration of sequencers of each ADC group (sequencer scan length, ...). Refer to function LL_ADC_REG_SetSequencerLength() and to function LL_ADC_INJ_SetSequencerLength().

Reference Manual to LL API cross reference:

- CR1 SCAN LL_ADC_GetSequencersScanMode

LL_ADC_REG_SetTriggerSource

Function name

__STATIC_INLINE void LL_ADC_REG_SetTriggerSource (ADC_TypeDef * ADCx, uint32_t TriggerSource)

Function description

Set ADC group regular conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line).

Parameters

- **ADCx:** ADC instance
- **TriggerSource:** This parameter can be one of the following values:
  - LL_ADC_REG_TRIG_SOFTWARE
  - LL_ADC_REG_TRIG_EXT_TIM1_CH1
  - LL_ADC_REG_TRIG_EXT_TIM1_CH2
  - LL_ADC_REG_TRIG_EXT_TIM1_CH3
  - LL_ADC_REG_TRIG_EXT_TIM2_CH2
  - LL_ADC_REG_TRIG_EXT_TIM2_CH3
  - LL_ADC_REG_TRIG_EXT_TIM2_CH4
  - LL_ADC_REG_TRIG_EXT_TIM2_TRGO
  - LL_ADC_REG_TRIG_EXT_TIM3_CH1
  - LL_ADC_REG_TRIG_EXT_TIM3_TRGO
  - LL_ADC_REG_TRIG_EXT_TIM4_CH4
  - LL_ADC_REG_TRIG_EXT_TIM5_CH1
  - LL_ADC_REG_TRIG_EXT_TIM5_CH2
  - LL_ADC_REG_TRIG_EXT_TIM5_CH3
  - LL_ADC_REG_TRIG_EXT_TIM8_CH1
  - LL_ADC_REG_TRIG_EXT_TIM8_TRGO
  - LL_ADC_REG_TRIG_EXT_EXTI_LINE11

Return values

- **None:**

Notes

- On this STM32 serie, setting of external trigger edge is performed using function LL_ADC_REG_StartConversionExtTrig().
- Availability of parameters of trigger sources from timer depends on timers availability on the selected device.

Reference Manual to LL API cross reference:

- CR2 EXTSEL LL_ADC_REG_SetTriggerSource
- CR2 EXTEN LL_ADC_REG_SetTriggerSource

LL_ADC_REG_GetTriggerSource

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_GetTriggerSource (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line).

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_REG_TRIG_SOFTWARE
  - LL_ADC_REG_TRIG_EXT_TIM1_CH1
  - LL_ADC_REG_TRIG_EXT_TIM1_CH2
  - LL_ADC_REG_TRIG_EXT_TIM1_CH3
  - LL_ADC_REG_TRIG_EXT_TIM2_CH2
  - LL_ADC_REG_TRIG_EXT_TIM2_CH3
  - LL_ADC_REG_TRIG_EXT_TIM2_CH4
  - LL_ADC_REG_TRIG_EXT_TIM2_TRGO
  - LL_ADC_REG_TRIG_EXT_TIM3_CH1
  - LL_ADC_REG_TRIG_EXT_TIM3_TRGO
  - LL_ADC_REG_TRIG_EXT_TIM4_CH4
  - LL_ADC_REG_TRIG_EXT_TIM5_CH1
  - LL_ADC_REG_TRIG_EXT_TIM5_CH2
  - LL_ADC_REG_TRIG_EXT_TIM5_CH3
  - LL_ADC_REG_TRIG_EXT_TIM8_CH1
  - LL_ADC_REG_TRIG_EXT_TIM8_TRGO
  - LL_ADC_REG_TRIG_EXT_EXTI_LINE11

Notes

- To determine whether group regular trigger source is internal (SW start) or external, without detail of which peripheral is selected as external trigger, (equivalent to "if(LL_ADC_REG_GetTriggerSource(ADC1) == LL_ADC_REG_TRIG_SOFTWARE)") use function LL_ADC_REG_IsTriggerSourceSWStart.
- Availability of parameters of trigger sources from timer depends on timers availability on the selected device.

Reference Manual to LL API cross reference:

- CR2 EXTSEL LL_ADC_REG_GetTriggerSource
- CR2 EXTEN LL_ADC_REG_GetTriggerSource

LL_ADC_REG_IsTriggerSourceSWStart

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_IsTriggerSourceSWStart (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion trigger source internal (SW start) or external.

Parameters

- **ADCx:** ADC instance

Return values

- **Value:** "0" if trigger source external trigger Value "1" if trigger source SW start.

Notes

- In case of group regular trigger source set to external trigger, to determine which peripheral is selected as external trigger, use function LL_ADC_REG_GetTriggerSource().

Reference Manual to LL API cross reference:

- CR2 EXTEN LL_ADC_REG_IsTriggerSourceSWStart

LL_ADC_REG_GetTriggerEdge

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_GetTriggerEdge (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion trigger polarity.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_REG_TRIG_EXT_RISING
  - LL_ADC_REG_TRIG_EXT_FALLING
  - LL_ADC_REG_TRIG_EXT_RISINGFALLING

Notes

- Applicable only for trigger source set to external trigger.
- On this STM32 serie, setting of external trigger edge is performed using function LL_ADC_REG_StartConversionExtTrig().

Reference Manual to LL API cross reference:

- CR2 EXTEN LL_ADC_REG_GetTriggerEdge

LL_ADC_REG_SetSequencerLength

Function name

__STATIC_INLINE void LL_ADC_REG_SetSequencerLength (ADC_TypeDef * ADCx, uint32_t SequencerNbRanks)

Function description

Set ADC group regular sequencer length and scan direction.

Parameters

- **ADCx:** ADC instance
- **SequencerNbRanks:** This parameter can be one of the following values:
  - LL_ADC_REG_SEQ_SCAN_DISABLE
  - LL_ADC_REG_SEQ_SCAN_ENABLE_2RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_3RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_4RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_5RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_6RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_7RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_8RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_9RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_10RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_11RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_12RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_13RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_14RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_15RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_16RANKS

Return values

- **None:**

Notes

- Description of ADC group regular sequencer features: For devices with sequencer fully configurable (function "LL_ADC_REG_SetSequencerRanks()" available): sequencer length and each rank affectation to a channel are configurable. This function performs configuration of: Sequence length: Number of ranks in the scan sequence.Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from rank 1 to rank n). Sequencer ranks are selected using function "LL_ADC_REG_SetSequencerRanks()". For devices with sequencer not fully configurable (function "LL_ADC_REG_SetSequencerChannels()" available): sequencer length and each rank affectation to a channel are defined by channel number. This function performs configuration of: Sequence length: Number of ranks in the scan sequence is defined by number of channels set in the sequence, rank of each channel is fixed by channel HW number. (channel 0 fixed on rank 0, channel 1 fixed on rank1, ...).Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from lowest channel number to highest channel number). Sequencer ranks are selected using function "LL_ADC_REG_SetSequencerChannels()".
- On this STM32 serie, group regular sequencer configuration is conditioned to ADC instance sequencer mode. If ADC instance sequencer mode is disabled, sequencers of all groups (group regular, group injected) can be configured but their execution is disabled (limited to rank 1). Refer to function LL_ADC_SetSequencersScanMode().
- Sequencer disabled is equivalent to sequencer of 1 rank: ADC conversion on only 1 channel.

Reference Manual to LL API cross reference:

- SQR1 L LL_ADC_REG_SetSequencerLength

LL_ADC_REG_GetSequencerLength

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_GetSequencerLength (ADC_TypeDef * ADCx)

Function description

Get ADC group regular sequencer length and scan direction.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_REG_SEQ_SCAN_DISABLE
  - LL_ADC_REG_SEQ_SCAN_ENABLE_2RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_3RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_4RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_5RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_6RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_7RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_8RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_9RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_10RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_11RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_12RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_13RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_14RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_15RANKS
  - LL_ADC_REG_SEQ_SCAN_ENABLE_16RANKS

Notes

- Description of ADC group regular sequencer features: For devices with sequencer fully configurable (function "LL_ADC_REG_SetSequencerRanks()" available): sequencer length and each rank affectation to a channel are configurable. This function retrieves: Sequence length: Number of ranks in the scan sequence.Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from rank 1 to rank n). Sequencer ranks are selected using function "LL_ADC_REG_SetSequencerRanks()". For devices with sequencer not fully configurable (function "LL_ADC_REG_SetSequencerChannels()" available): sequencer length and each rank affectation to a channel are defined by channel number. This function retrieves: Sequence length: Number of ranks in the scan sequence is defined by number of channels set in the sequence, rank of each channel is fixed by channel HW number. (channel 0 fixed on rank 0, channel 1 fixed on rank1, ...).Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from lowest channel number to highest channel number). Sequencer ranks are selected using function "LL_ADC_REG_SetSequencerChannels()".
- On this STM32 serie, group regular sequencer configuration is conditioned to ADC instance sequencer mode. If ADC instance sequencer mode is disabled, sequencers of all groups (group regular, group injected) can be configured but their execution is disabled (limited to rank 1). Refer to function LL_ADC_SetSequencersScanMode().
- Sequencer disabled is equivalent to sequencer of 1 rank: ADC conversion on only 1 channel.

Reference Manual to LL API cross reference:

- SQR1 L LL_ADC_REG_SetSequencerLength

LL_ADC_REG_SetSequencerDiscont

Function name

__STATIC_INLINE void LL_ADC_REG_SetSequencerDiscont (ADC_TypeDef * ADCx, uint32_t SeqDiscont)

Function description

Set ADC group regular sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks.

Parameters

- **ADCx:** ADC instance
- **SeqDiscont:** This parameter can be one of the following values:
  - LL_ADC_REG_SEQ_DISCONT_DISABLE
  - LL_ADC_REG_SEQ_DISCONT_1RANK
  - LL_ADC_REG_SEQ_DISCONT_2RANKS
  - LL_ADC_REG_SEQ_DISCONT_3RANKS
  - LL_ADC_REG_SEQ_DISCONT_4RANKS
  - LL_ADC_REG_SEQ_DISCONT_5RANKS
  - LL_ADC_REG_SEQ_DISCONT_6RANKS
  - LL_ADC_REG_SEQ_DISCONT_7RANKS
  - LL_ADC_REG_SEQ_DISCONT_8RANKS

Return values

- **None:**

Notes

- It is not possible to enable both ADC group regular continuous mode and sequencer discontinuous mode.
- It is not possible to enable both ADC auto-injected mode and ADC group regular sequencer discontinuous mode.

Reference Manual to LL API cross reference:

- CR1 DISCEN LL_ADC_REG_SetSequencerDiscont
- CR1 DISCNUM LL_ADC_REG_SetSequencerDiscont

LL_ADC_REG_GetSequencerDiscont

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_GetSequencerDiscont (ADC_TypeDef * ADCx)

Function description

Get ADC group regular sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_REG_SEQ_DISCONT_DISABLE
  - LL_ADC_REG_SEQ_DISCONT_1RANK
  - LL_ADC_REG_SEQ_DISCONT_2RANKS
  - LL_ADC_REG_SEQ_DISCONT_3RANKS
  - LL_ADC_REG_SEQ_DISCONT_4RANKS
  - LL_ADC_REG_SEQ_DISCONT_5RANKS
  - LL_ADC_REG_SEQ_DISCONT_6RANKS
  - LL_ADC_REG_SEQ_DISCONT_7RANKS
  - LL_ADC_REG_SEQ_DISCONT_8RANKS

Reference Manual to LL API cross reference:

- CR1 DISCEN LL_ADC_REG_GetSequencerDiscont
- CR1 DISCNUM LL_ADC_REG_GetSequencerDiscont

LL_ADC_REG_SetSequencerRanks

Function name

__STATIC_INLINE void LL_ADC_REG_SetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank, uint32_t Channel)

Function description

Set ADC group regular sequence: channel on the selected scan sequence rank.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_REG_RANK_1
  - LL_ADC_REG_RANK_2
  - LL_ADC_REG_RANK_3
  - LL_ADC_REG_RANK_4
  - LL_ADC_REG_RANK_5
  - LL_ADC_REG_RANK_6
  - LL_ADC_REG_RANK_7
  - LL_ADC_REG_RANK_8
  - LL_ADC_REG_RANK_9
  - LL_ADC_REG_RANK_10
  - LL_ADC_REG_RANK_11
  - LL_ADC_REG_RANK_12
  - LL_ADC_REG_RANK_13
  - LL_ADC_REG_RANK_14
  - LL_ADC_REG_RANK_15
  - LL_ADC_REG_RANK_16
- **Channel:** This parameter can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

  (1) On STM32F4, parameter available only on ADC instance: ADC1.
- (2) On devices STM32F42x and STM32F43x, limitation: this internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.

Return values

- **None:**

Notes

- This function performs configuration of: Channels ordering into each rank of scan sequence: whatever channel can be placed into whatever rank.
- On this STM32 serie, ADC group regular sequencer is fully configurable: sequencer length and each rank affectation to a channel are configurable. Refer to description of function LL_ADC_REG_SetSequencerLength().
- Depending on devices and packages, some channels may not be available. Refer to device datasheet for channels availability.
- On this STM32 serie, to measure internal channels (VrefInt, TempSensor, ...), measurement paths to internal channels must be enabled separately. This can be done using function LL_ADC_SetCommonPathInternalCh().

Reference Manual to LL API cross reference:

- SQR3 SQ1 LL_ADC_REG_SetSequencerRanks
- SQR3 SQ2 LL_ADC_REG_SetSequencerRanks
- SQR3 SQ3 LL_ADC_REG_SetSequencerRanks
- SQR3 SQ4 LL_ADC_REG_SetSequencerRanks
- SQR3 SQ5 LL_ADC_REG_SetSequencerRanks
- SQR3 SQ6 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ7 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ8 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ9 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ10 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ11 LL_ADC_REG_SetSequencerRanks
- SQR2 SQ12 LL_ADC_REG_SetSequencerRanks
- SQR1 SQ13 LL_ADC_REG_SetSequencerRanks
- SQR1 SQ14 LL_ADC_REG_SetSequencerRanks
- SQR1 SQ15 LL_ADC_REG_SetSequencerRanks
- SQR1 SQ16 LL_ADC_REG_SetSequencerRanks

LL_ADC_REG_GetSequencerRanks

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_GetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank)

Function description

Get ADC group regular sequence: channel on the selected scan sequence rank.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_REG_RANK_1
  - LL_ADC_REG_RANK_2
  - LL_ADC_REG_RANK_3
  - LL_ADC_REG_RANK_4
  - LL_ADC_REG_RANK_5
  - LL_ADC_REG_RANK_6
  - LL_ADC_REG_RANK_7
  - LL_ADC_REG_RANK_8
  - LL_ADC_REG_RANK_9
  - LL_ADC_REG_RANK_10
  - LL_ADC_REG_RANK_11
  - LL_ADC_REG_RANK_12
  - LL_ADC_REG_RANK_13
  - LL_ADC_REG_RANK_14
  - LL_ADC_REG_RANK_15
  - LL_ADC_REG_RANK_16

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

  (1) On STM32F4, parameter available only on ADC instance: ADC1.
- (2) On devices STM32F42x and STM32F43x, limitation: this internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.
- (1) For ADC channel read back from ADC register, comparison with internal channel parameter to be done using helper macro __LL_ADC_CHANNEL_INTERNAL_TO_EXTERNAL().

Notes

- On this STM32 serie, ADC group regular sequencer is fully configurable: sequencer length and each rank affectation to a channel are configurable. Refer to description of function LL_ADC_REG_SetSequencerLength().
- Depending on devices and packages, some channels may not be available. Refer to device datasheet for channels availability.
- Usage of the returned channel number: To reinject this channel into another function LL_ADC_xxx: the returned channel number is only partly formatted on definition of literals LL_ADC_CHANNEL_x. Therefore, it has to be compared with parts of literals LL_ADC_CHANNEL_x or using helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB(). Then the selected literal LL_ADC_CHANNEL_x can be used as parameter for another function.To get the channel number in decimal format: process the returned value with the helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB().

Reference Manual to LL API cross reference:

- SQR3 SQ1 LL_ADC_REG_GetSequencerRanks
- SQR3 SQ2 LL_ADC_REG_GetSequencerRanks
- SQR3 SQ3 LL_ADC_REG_GetSequencerRanks
- SQR3 SQ4 LL_ADC_REG_GetSequencerRanks
- SQR3 SQ5 LL_ADC_REG_GetSequencerRanks
- SQR3 SQ6 LL_ADC_REG_GetSequencerRanks
- SQR2 SQ7 LL_ADC_REG_GetSequencerRanks
- SQR2 SQ8 LL_ADC_REG_GetSequencerRanks
- SQR2 SQ9 LL_ADC_REG_GetSequencerRanks
- SQR2 SQ10 LL_ADC_REG_GetSequencerRanks
- SQR2 SQ11 LL_ADC_REG_GetSequencerRanks
- SQR2 SQ12 LL_ADC_REG_GetSequencerRanks
- SQR1 SQ13 LL_ADC_REG_GetSequencerRanks
- SQR1 SQ14 LL_ADC_REG_GetSequencerRanks
- SQR1 SQ15 LL_ADC_REG_GetSequencerRanks
- SQR1 SQ16 LL_ADC_REG_GetSequencerRanks

LL_ADC_REG_SetContinuousMode

Function name

__STATIC_INLINE void LL_ADC_REG_SetContinuousMode (ADC_TypeDef * ADCx, uint32_t Continuous)

Function description

Set ADC continuous conversion mode on ADC group regular.

Parameters

- **ADCx:** ADC instance
- **Continuous:** This parameter can be one of the following values:
  - LL_ADC_REG_CONV_SINGLE
  - LL_ADC_REG_CONV_CONTINUOUS

Return values

- **None:**

Notes

- Description of ADC continuous conversion mode: single mode: one conversion per triggercontinuous mode: after the first trigger, following conversions launched successively automatically.
- It is not possible to enable both ADC group regular continuous mode and sequencer discontinuous mode.

Reference Manual to LL API cross reference:

- CR2 CONT LL_ADC_REG_SetContinuousMode

LL_ADC_REG_GetContinuousMode

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_GetContinuousMode (ADC_TypeDef * ADCx)

Function description

Get ADC continuous conversion mode on ADC group regular.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_REG_CONV_SINGLE
  - LL_ADC_REG_CONV_CONTINUOUS

Notes

- Description of ADC continuous conversion mode: single mode: one conversion per triggercontinuous mode: after the first trigger, following conversions launched successively automatically.

Reference Manual to LL API cross reference:

- CR2 CONT LL_ADC_REG_GetContinuousMode

LL_ADC_REG_SetDMATransfer

Function name

__STATIC_INLINE void LL_ADC_REG_SetDMATransfer (ADC_TypeDef * ADCx, uint32_t DMATransfer)

Function description

Set ADC group regular conversion data transfer: no transfer or transfer by DMA, and DMA requests mode.

Parameters

- **ADCx:** ADC instance
- **DMATransfer:** This parameter can be one of the following values:
  - LL_ADC_REG_DMA_TRANSFER_NONE
  - LL_ADC_REG_DMA_TRANSFER_LIMITED
  - LL_ADC_REG_DMA_TRANSFER_UNLIMITED

Return values

- **None:**

Notes

- If transfer by DMA selected, specifies the DMA requests mode: Limited mode (One shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular.Unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transfers (number of ADC conversions). This ADC mode is intended to be used with DMA mode circular.
- If ADC DMA requests mode is set to unlimited and DMA is set to mode non-circular: when DMA transfers size will be reached, DMA will stop transfers of ADC conversions data ADC will raise an overrun error (overrun flag and interruption if enabled).
- For devices with several ADC instances: ADC multimode DMA settings are available using function LL_ADC_SetMultiDMATransfer().
- To configure DMA source address (peripheral address), use function LL_ADC_DMA_GetRegAddr().

Reference Manual to LL API cross reference:

- CR2 DMA LL_ADC_REG_SetDMATransfer
- CR2 DDS LL_ADC_REG_SetDMATransfer

LL_ADC_REG_GetDMATransfer

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_GetDMATransfer (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion data transfer: no transfer or transfer by DMA, and DMA requests mode.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_REG_DMA_TRANSFER_NONE
  - LL_ADC_REG_DMA_TRANSFER_LIMITED
  - LL_ADC_REG_DMA_TRANSFER_UNLIMITED

Notes

- If transfer by DMA selected, specifies the DMA requests mode: Limited mode (One shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular.Unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transfers (number of ADC conversions). This ADC mode is intended to be used with DMA mode circular.
- If ADC DMA requests mode is set to unlimited and DMA is set to mode non-circular: when DMA transfers size will be reached, DMA will stop transfers of ADC conversions data ADC will raise an overrun error (overrun flag and interruption if enabled).
- For devices with several ADC instances: ADC multimode DMA settings are available using function LL_ADC_GetMultiDMATransfer().
- To configure DMA source address (peripheral address), use function LL_ADC_DMA_GetRegAddr().

Reference Manual to LL API cross reference:

- CR2 DMA LL_ADC_REG_GetDMATransfer
- CR2 DDS LL_ADC_REG_GetDMATransfer

LL_ADC_REG_SetFlagEndOfConversion

Function name

__STATIC_INLINE void LL_ADC_REG_SetFlagEndOfConversion (ADC_TypeDef * ADCx, uint32_t EocSelection)

Function description

Specify which ADC flag between EOC (end of unitary conversion) or EOS (end of sequence conversions) is used to indicate the end of conversion.

Parameters

- **ADCx:** ADC instance
- **EocSelection:** This parameter can be one of the following values:
  - LL_ADC_REG_FLAG_EOC_SEQUENCE_CONV
  - LL_ADC_REG_FLAG_EOC_UNITARY_CONV

Return values

- **None:**

Notes

- This feature is aimed to be set when using ADC with programming model by polling or interruption (programming model by DMA usually uses DMA interruptions to indicate end of conversion and data transfer).
- For ADC group injected, end of conversion (flag&IT) is raised only at the end of the sequence.

Reference Manual to LL API cross reference:

- CR2 EOCS LL_ADC_REG_SetFlagEndOfConversion

LL_ADC_REG_GetFlagEndOfConversion

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_GetFlagEndOfConversion (ADC_TypeDef * ADCx)

Function description

Get which ADC flag between EOC (end of unitary conversion) or EOS (end of sequence conversions) is used to indicate the end of conversion.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_REG_FLAG_EOC_SEQUENCE_CONV
  - LL_ADC_REG_FLAG_EOC_UNITARY_CONV

Reference Manual to LL API cross reference:

- CR2 EOCS LL_ADC_REG_GetFlagEndOfConversion

LL_ADC_INJ_SetTriggerSource

Function name

__STATIC_INLINE void LL_ADC_INJ_SetTriggerSource (ADC_TypeDef * ADCx, uint32_t TriggerSource)

Function description

Set ADC group injected conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line).

Parameters

- **ADCx:** ADC instance
- **TriggerSource:** This parameter can be one of the following values:
  - LL_ADC_INJ_TRIG_SOFTWARE
  - LL_ADC_INJ_TRIG_EXT_TIM1_CH4
  - LL_ADC_INJ_TRIG_EXT_TIM1_TRGO
  - LL_ADC_INJ_TRIG_EXT_TIM2_CH1
  - LL_ADC_INJ_TRIG_EXT_TIM2_TRGO
  - LL_ADC_INJ_TRIG_EXT_TIM3_CH2
  - LL_ADC_INJ_TRIG_EXT_TIM3_CH4
  - LL_ADC_INJ_TRIG_EXT_TIM4_CH1
  - LL_ADC_INJ_TRIG_EXT_TIM4_CH2
  - LL_ADC_INJ_TRIG_EXT_TIM4_CH3
  - LL_ADC_INJ_TRIG_EXT_TIM4_TRGO
  - LL_ADC_INJ_TRIG_EXT_TIM5_CH4
  - LL_ADC_INJ_TRIG_EXT_TIM5_TRGO
  - LL_ADC_INJ_TRIG_EXT_TIM8_CH2
  - LL_ADC_INJ_TRIG_EXT_TIM8_CH3
  - LL_ADC_INJ_TRIG_EXT_TIM8_CH4
  - LL_ADC_INJ_TRIG_EXT_EXTI_LINE15

Return values

- **None:**

Notes

- On this STM32 serie, setting of external trigger edge is performed using function LL_ADC_INJ_StartConversionExtTrig().
- Availability of parameters of trigger sources from timer depends on timers availability on the selected device.

Reference Manual to LL API cross reference:

- CR2 JEXTSEL LL_ADC_INJ_SetTriggerSource
- CR2 JEXTEN LL_ADC_INJ_SetTriggerSource

LL_ADC_INJ_GetTriggerSource

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_GetTriggerSource (ADC_TypeDef * ADCx)

Function description

Get ADC group injected conversion trigger source: internal (SW start) or from external IP (timer event, external interrupt line).

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_INJ_TRIG_SOFTWARE
  - LL_ADC_INJ_TRIG_EXT_TIM1_CH4
  - LL_ADC_INJ_TRIG_EXT_TIM1_TRGO
  - LL_ADC_INJ_TRIG_EXT_TIM2_CH1
  - LL_ADC_INJ_TRIG_EXT_TIM2_TRGO
  - LL_ADC_INJ_TRIG_EXT_TIM3_CH2
  - LL_ADC_INJ_TRIG_EXT_TIM3_CH4
  - LL_ADC_INJ_TRIG_EXT_TIM4_CH1
  - LL_ADC_INJ_TRIG_EXT_TIM4_CH2
  - LL_ADC_INJ_TRIG_EXT_TIM4_CH3
  - LL_ADC_INJ_TRIG_EXT_TIM4_TRGO
  - LL_ADC_INJ_TRIG_EXT_TIM5_CH4
  - LL_ADC_INJ_TRIG_EXT_TIM5_TRGO
  - LL_ADC_INJ_TRIG_EXT_TIM8_CH2
  - LL_ADC_INJ_TRIG_EXT_TIM8_CH3
  - LL_ADC_INJ_TRIG_EXT_TIM8_CH4
  - LL_ADC_INJ_TRIG_EXT_EXTI_LINE15

Notes

- To determine whether group injected trigger source is internal (SW start) or external, without detail of which peripheral is selected as external trigger, (equivalent to "if(LL_ADC_INJ_GetTriggerSource(ADC1) == LL_ADC_INJ_TRIG_SOFTWARE)") use function LL_ADC_INJ_IsTriggerSourceSWStart.
- Availability of parameters of trigger sources from timer depends on timers availability on the selected device.

Reference Manual to LL API cross reference:

- CR2 JEXTSEL LL_ADC_INJ_GetTriggerSource
- CR2 JEXTEN LL_ADC_INJ_GetTriggerSource

LL_ADC_INJ_IsTriggerSourceSWStart

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_IsTriggerSourceSWStart (ADC_TypeDef * ADCx)

Function description

Get ADC group injected conversion trigger source internal (SW start) or external.

Parameters

- **ADCx:** ADC instance

Return values

- **Value:** "0" if trigger source external trigger Value "1" if trigger source SW start.

Notes

- In case of group injected trigger source set to external trigger, to determine which peripheral is selected as external trigger, use function LL_ADC_INJ_GetTriggerSource.

Reference Manual to LL API cross reference:

- CR2 JEXTEN LL_ADC_INJ_IsTriggerSourceSWStart

LL_ADC_INJ_GetTriggerEdge

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_GetTriggerEdge (ADC_TypeDef * ADCx)

Function description

Get ADC group injected conversion trigger polarity.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_INJ_TRIG_EXT_RISING
  - LL_ADC_INJ_TRIG_EXT_FALLING
  - LL_ADC_INJ_TRIG_EXT_RISINGFALLING

Reference Manual to LL API cross reference:

- CR2 JEXTEN LL_ADC_INJ_GetTriggerEdge

LL_ADC_INJ_SetSequencerLength

Function name

__STATIC_INLINE void LL_ADC_INJ_SetSequencerLength (ADC_TypeDef * ADCx, uint32_t SequencerNbRanks)

Function description

Set ADC group injected sequencer length and scan direction.

Parameters

- **ADCx:** ADC instance
- **SequencerNbRanks:** This parameter can be one of the following values:
  - LL_ADC_INJ_SEQ_SCAN_DISABLE
  - LL_ADC_INJ_SEQ_SCAN_ENABLE_2RANKS
  - LL_ADC_INJ_SEQ_SCAN_ENABLE_3RANKS
  - LL_ADC_INJ_SEQ_SCAN_ENABLE_4RANKS

Return values

- **None:**

Notes

- This function performs configuration of: Sequence length: Number of ranks in the scan sequence.Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from rank 1 to rank n).
- On this STM32 serie, group injected sequencer configuration is conditioned to ADC instance sequencer mode. If ADC instance sequencer mode is disabled, sequencers of all groups (group regular, group injected) can be configured but their execution is disabled (limited to rank 1). Refer to function LL_ADC_SetSequencersScanMode().
- Sequencer disabled is equivalent to sequencer of 1 rank: ADC conversion on only 1 channel.

Reference Manual to LL API cross reference:

- JSQR JL LL_ADC_INJ_SetSequencerLength

LL_ADC_INJ_GetSequencerLength

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_GetSequencerLength (ADC_TypeDef * ADCx)

Function description

Get ADC group injected sequencer length and scan direction.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_INJ_SEQ_SCAN_DISABLE
  - LL_ADC_INJ_SEQ_SCAN_ENABLE_2RANKS
  - LL_ADC_INJ_SEQ_SCAN_ENABLE_3RANKS
  - LL_ADC_INJ_SEQ_SCAN_ENABLE_4RANKS

Notes

- This function retrieves: Sequence length: Number of ranks in the scan sequence.Sequence direction: Unless specified in parameters, sequencer scan direction is forward (from rank 1 to rank n).
- On this STM32 serie, group injected sequencer configuration is conditioned to ADC instance sequencer mode. If ADC instance sequencer mode is disabled, sequencers of all groups (group regular, group injected) can be configured but their execution is disabled (limited to rank 1). Refer to function LL_ADC_SetSequencersScanMode().
- Sequencer disabled is equivalent to sequencer of 1 rank: ADC conversion on only 1 channel.

Reference Manual to LL API cross reference:

- JSQR JL LL_ADC_INJ_GetSequencerLength

LL_ADC_INJ_SetSequencerDiscont

Function name

__STATIC_INLINE void LL_ADC_INJ_SetSequencerDiscont (ADC_TypeDef * ADCx, uint32_t SeqDiscont)

Function description

Set ADC group injected sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks.

Parameters

- **ADCx:** ADC instance
- **SeqDiscont:** This parameter can be one of the following values:
  - LL_ADC_INJ_SEQ_DISCONT_DISABLE
  - LL_ADC_INJ_SEQ_DISCONT_1RANK

Return values

- **None:**

Notes

- It is not possible to enable both ADC group injected auto-injected mode and sequencer discontinuous mode.

Reference Manual to LL API cross reference:

- CR1 DISCEN LL_ADC_INJ_SetSequencerDiscont

LL_ADC_INJ_GetSequencerDiscont

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_GetSequencerDiscont (ADC_TypeDef * ADCx)

Function description

Get ADC group injected sequencer discontinuous mode: sequence subdivided and scan conversions interrupted every selected number of ranks.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_INJ_SEQ_DISCONT_DISABLE
  - LL_ADC_INJ_SEQ_DISCONT_1RANK

Reference Manual to LL API cross reference:

- CR1 DISCEN LL_ADC_REG_GetSequencerDiscont

LL_ADC_INJ_SetSequencerRanks

Function name

__STATIC_INLINE void LL_ADC_INJ_SetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank, uint32_t Channel)

Function description

Set ADC group injected sequence: channel on the selected sequence rank.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4
- **Channel:** This parameter can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

  (1) On STM32F4, parameter available only on ADC instance: ADC1.
- (2) On devices STM32F42x and STM32F43x, limitation: this internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.

Return values

- **None:**

Notes

- Depending on devices and packages, some channels may not be available. Refer to device datasheet for channels availability.
- On this STM32 serie, to measure internal channels (VrefInt, TempSensor, ...), measurement paths to internal channels must be enabled separately. This can be done using function LL_ADC_SetCommonPathInternalCh().

Reference Manual to LL API cross reference:

- JSQR JSQ1 LL_ADC_INJ_SetSequencerRanks
- JSQR JSQ2 LL_ADC_INJ_SetSequencerRanks
- JSQR JSQ3 LL_ADC_INJ_SetSequencerRanks
- JSQR JSQ4 LL_ADC_INJ_SetSequencerRanks

LL_ADC_INJ_GetSequencerRanks

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_GetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank)

Function description

Get ADC group injected sequence: channel on the selected sequence rank.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

  (1) On STM32F4, parameter available only on ADC instance: ADC1.
- (2) On devices STM32F42x and STM32F43x, limitation: this internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.
- (1) For ADC channel read back from ADC register, comparison with internal channel parameter to be done using helper macro __LL_ADC_CHANNEL_INTERNAL_TO_EXTERNAL().

Notes

- Depending on devices and packages, some channels may not be available. Refer to device datasheet for channels availability.
- Usage of the returned channel number: To reinject this channel into another function LL_ADC_xxx: the returned channel number is only partly formatted on definition of literals LL_ADC_CHANNEL_x. Therefore, it has to be compared with parts of literals LL_ADC_CHANNEL_x or using helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB(). Then the selected literal LL_ADC_CHANNEL_x can be used as parameter for another function.To get the channel number in decimal format: process the returned value with the helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB().

Reference Manual to LL API cross reference:

- JSQR JSQ1 LL_ADC_INJ_SetSequencerRanks
- JSQR JSQ2 LL_ADC_INJ_SetSequencerRanks
- JSQR JSQ3 LL_ADC_INJ_SetSequencerRanks
- JSQR JSQ4 LL_ADC_INJ_SetSequencerRanks

LL_ADC_INJ_SetTrigAuto

Function name

__STATIC_INLINE void LL_ADC_INJ_SetTrigAuto (ADC_TypeDef * ADCx, uint32_t TrigAuto)

Function description

Set ADC group injected conversion trigger: independent or from ADC group regular.

Parameters

- **ADCx:** ADC instance
- **TrigAuto:** This parameter can be one of the following values:
  - LL_ADC_INJ_TRIG_INDEPENDENT
  - LL_ADC_INJ_TRIG_FROM_GRP_REGULAR

Return values

- **None:**

Notes

- This mode can be used to extend number of data registers updated after one ADC conversion trigger and with data permanently kept (not erased by successive conversions of scan of ADC sequencer ranks), up to 5 data registers: 1 data register on ADC group regular, 4 data registers on ADC group injected.
- If ADC group injected injected trigger source is set to an external trigger, this feature must be must be set to independent trigger. ADC group injected automatic trigger is compliant only with group injected trigger source set to SW start, without any further action on ADC group injected conversion start or stop: in this case, ADC group injected is controlled only from ADC group regular.
- It is not possible to enable both ADC group injected auto-injected mode and sequencer discontinuous mode.

Reference Manual to LL API cross reference:

- CR1 JAUTO LL_ADC_INJ_SetTrigAuto

LL_ADC_INJ_GetTrigAuto

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_GetTrigAuto (ADC_TypeDef * ADCx)

Function description

Get ADC group injected conversion trigger: independent or from ADC group regular.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_INJ_TRIG_INDEPENDENT
  - LL_ADC_INJ_TRIG_FROM_GRP_REGULAR

Reference Manual to LL API cross reference:

- CR1 JAUTO LL_ADC_INJ_GetTrigAuto

LL_ADC_INJ_SetOffset

Function name

__STATIC_INLINE void LL_ADC_INJ_SetOffset (ADC_TypeDef * ADCx, uint32_t Rank, uint32_t OffsetLevel)

Function description

Set ADC group injected offset.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4
- **OffsetLevel:** Value between Min_Data=0x000 and Max_Data=0xFFF

Return values

- **None:**

Notes

- It sets: ADC group injected rank to which the offset programmed will be appliedOffset level (offset to be subtracted from the raw converted data). Caution: Offset format is dependent to ADC resolution: offset has to be left-aligned on bit 11, the LSB (right bits) are set to 0.
- Offset cannot be enabled or disabled. To emulate offset disabled, set an offset value equal to 0.

Reference Manual to LL API cross reference:

- JOFR1 JOFFSET1 LL_ADC_INJ_SetOffset
- JOFR2 JOFFSET2 LL_ADC_INJ_SetOffset
- JOFR3 JOFFSET3 LL_ADC_INJ_SetOffset
- JOFR4 JOFFSET4 LL_ADC_INJ_SetOffset

LL_ADC_INJ_GetOffset

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_GetOffset (ADC_TypeDef * ADCx, uint32_t Rank)

Function description

Get ADC group injected offset.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4

Return values

- **Value:** between Min_Data=0x000 and Max_Data=0xFFF

Notes

- It gives offset level (offset to be subtracted from the raw converted data). Caution: Offset format is dependent to ADC resolution: offset has to be left-aligned on bit 11, the LSB (right bits) are set to 0.

Reference Manual to LL API cross reference:

- JOFR1 JOFFSET1 LL_ADC_INJ_GetOffset
- JOFR2 JOFFSET2 LL_ADC_INJ_GetOffset
- JOFR3 JOFFSET3 LL_ADC_INJ_GetOffset
- JOFR4 JOFFSET4 LL_ADC_INJ_GetOffset

LL_ADC_SetChannelSamplingTime

Function name

__STATIC_INLINE void LL_ADC_SetChannelSamplingTime (ADC_TypeDef * ADCx, uint32_t Channel, uint32_t SamplingTime)

Function description

Set sampling time of the selected ADC channel Unit: ADC clock cycles.

Parameters

- **ADCx:** ADC instance
- **Channel:** This parameter can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

  (1) On STM32F4, parameter available only on ADC instance: ADC1.
- (2) On devices STM32F42x and STM32F43x, limitation: this internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.
- **SamplingTime:** This parameter can be one of the following values:
  - LL_ADC_SAMPLINGTIME_3CYCLES
  - LL_ADC_SAMPLINGTIME_15CYCLES
  - LL_ADC_SAMPLINGTIME_28CYCLES
  - LL_ADC_SAMPLINGTIME_56CYCLES
  - LL_ADC_SAMPLINGTIME_84CYCLES
  - LL_ADC_SAMPLINGTIME_112CYCLES
  - LL_ADC_SAMPLINGTIME_144CYCLES
  - LL_ADC_SAMPLINGTIME_480CYCLES

Return values

- **None:**

Notes

- On this device, sampling time is on channel scope: independently of channel mapped on ADC group regular or injected.
- In case of internal channel (VrefInt, TempSensor, ...) to be converted: sampling time constraints must be respected (sampling time can be adjusted in function of ADC clock frequency and sampling time setting). Refer to device datasheet for timings values (parameters TS_vrefint, TS_temp, ...).
- Conversion time is the addition of sampling time and processing time. Refer to reference manual for ADC processing time of this STM32 serie.
- In case of ADC conversion of internal channel (VrefInt, temperature sensor, ...), a sampling time minimum value is required. Refer to device datasheet.

Reference Manual to LL API cross reference:

- SMPR1 SMP18 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP17 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP16 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP15 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP14 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP13 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP12 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP11 LL_ADC_SetChannelSamplingTime
- SMPR1 SMP10 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP9 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP8 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP7 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP6 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP5 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP4 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP3 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP2 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP1 LL_ADC_SetChannelSamplingTime
- SMPR2 SMP0 LL_ADC_SetChannelSamplingTime

LL_ADC_GetChannelSamplingTime

Function name

__STATIC_INLINE uint32_t LL_ADC_GetChannelSamplingTime (ADC_TypeDef * ADCx, uint32_t Channel)

Function description

Get sampling time of the selected ADC channel Unit: ADC clock cycles.

Parameters

- **ADCx:** ADC instance
- **Channel:** This parameter can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

  (1) On STM32F4, parameter available only on ADC instance: ADC1.
- (2) On devices STM32F42x and STM32F43x, limitation: this internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_SAMPLINGTIME_3CYCLES
  - LL_ADC_SAMPLINGTIME_15CYCLES
  - LL_ADC_SAMPLINGTIME_28CYCLES
  - LL_ADC_SAMPLINGTIME_56CYCLES
  - LL_ADC_SAMPLINGTIME_84CYCLES
  - LL_ADC_SAMPLINGTIME_112CYCLES
  - LL_ADC_SAMPLINGTIME_144CYCLES
  - LL_ADC_SAMPLINGTIME_480CYCLES

Notes

- On this device, sampling time is on channel scope: independently of channel mapped on ADC group regular or injected.
- Conversion time is the addition of sampling time and processing time. Refer to reference manual for ADC processing time of this STM32 serie.

Reference Manual to LL API cross reference:

- SMPR1 SMP18 LL_ADC_GetChannelSamplingTime
- SMPR1 SMP17 LL_ADC_GetChannelSamplingTime
- SMPR1 SMP16 LL_ADC_GetChannelSamplingTime
- SMPR1 SMP15 LL_ADC_GetChannelSamplingTime
- SMPR1 SMP14 LL_ADC_GetChannelSamplingTime
- SMPR1 SMP13 LL_ADC_GetChannelSamplingTime
- SMPR1 SMP12 LL_ADC_GetChannelSamplingTime
- SMPR1 SMP11 LL_ADC_GetChannelSamplingTime
- SMPR1 SMP10 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP9 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP8 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP7 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP6 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP5 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP4 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP3 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP2 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP1 LL_ADC_GetChannelSamplingTime
- SMPR2 SMP0 LL_ADC_GetChannelSamplingTime

LL_ADC_SetAnalogWDMonitChannels

Function name

__STATIC_INLINE void LL_ADC_SetAnalogWDMonitChannels (ADC_TypeDef * ADCx, uint32_t AWDChannelGroup)

Function description

Set ADC analog watchdog monitored channels: a single channel or all channels, on ADC groups regular and-or injected.

Parameters

- **ADCx:** ADC instance

- **AWDChannelGroup:** This parameter can be one of the following values:
  - LL_ADC_AWD_DISABLE
  - LL_ADC_AWD_ALL_CHANNELS_REG
  - LL_ADC_AWD_ALL_CHANNELS_INJ
  - LL_ADC_AWD_ALL_CHANNELS_REG_INJ
  - LL_ADC_AWD_CHANNEL_0_REG
  - LL_ADC_AWD_CHANNEL_0_INJ
  - LL_ADC_AWD_CHANNEL_0_REG_INJ
  - LL_ADC_AWD_CHANNEL_1_REG
  - LL_ADC_AWD_CHANNEL_1_INJ
  - LL_ADC_AWD_CHANNEL_1_REG_INJ
  - LL_ADC_AWD_CHANNEL_2_REG
  - LL_ADC_AWD_CHANNEL_2_INJ
  - LL_ADC_AWD_CHANNEL_2_REG_INJ
  - LL_ADC_AWD_CHANNEL_3_REG
  - LL_ADC_AWD_CHANNEL_3_INJ
  - LL_ADC_AWD_CHANNEL_3_REG_INJ
  - LL_ADC_AWD_CHANNEL_4_REG
  - LL_ADC_AWD_CHANNEL_4_INJ
  - LL_ADC_AWD_CHANNEL_4_REG_INJ
  - LL_ADC_AWD_CHANNEL_5_REG
  - LL_ADC_AWD_CHANNEL_5_INJ
  - LL_ADC_AWD_CHANNEL_5_REG_INJ
  - LL_ADC_AWD_CHANNEL_6_REG
  - LL_ADC_AWD_CHANNEL_6_INJ
  - LL_ADC_AWD_CHANNEL_6_REG_INJ
  - LL_ADC_AWD_CHANNEL_7_REG
  - LL_ADC_AWD_CHANNEL_7_INJ
  - LL_ADC_AWD_CHANNEL_7_REG_INJ
  - LL_ADC_AWD_CHANNEL_8_REG
  - LL_ADC_AWD_CHANNEL_8_INJ
  - LL_ADC_AWD_CHANNEL_8_REG_INJ
  - LL_ADC_AWD_CHANNEL_9_REG
  - LL_ADC_AWD_CHANNEL_9_INJ
  - LL_ADC_AWD_CHANNEL_9_REG_INJ
  - LL_ADC_AWD_CHANNEL_10_REG
  - LL_ADC_AWD_CHANNEL_10_INJ
  - LL_ADC_AWD_CHANNEL_10_REG_INJ
  - LL_ADC_AWD_CHANNEL_11_REG
  - LL_ADC_AWD_CHANNEL_11_INJ
  - LL_ADC_AWD_CHANNEL_11_REG_INJ
  - LL_ADC_AWD_CHANNEL_12_REG
  - LL_ADC_AWD_CHANNEL_12_INJ
  - LL_ADC_AWD_CHANNEL_12_REG_INJ
  - LL_ADC_AWD_CHANNEL_13_REG
  - LL_ADC_AWD_CHANNEL_13_INJ
  - LL_ADC_AWD_CHANNEL_13_REG_INJ
  - LL_ADC_AWD_CHANNEL_14_REG
  - LL_ADC_AWD_CHANNEL_14_INJ
  - LL_ADC_AWD_CHANNEL_14_REG_INJ
  - LL_ADC_AWD_CHANNEL_15_REG
  - LL_ADC_AWD_CHANNEL_15_INJ
  - LL_ADC_AWD_CHANNEL_15_REG_INJ

- (2) On devices STM32F42x and STM32F43x, limitation: this internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.

Return values

- **None:**

Notes

- Once monitored channels are selected, analog watchdog is enabled.
- In case of need to define a single channel to monitor with analog watchdog from sequencer channel definition, use helper macro __LL_ADC_ANALOGWD_CHANNEL_GROUP().
- On this STM32 serie, there is only 1 kind of analog watchdog instance: AWD standard (instance AWD1): channels monitored: can monitor 1 channel or all channels.groups monitored: ADC groups regular and-or injected.resolution: resolution is not limited (corresponds to ADC resolution configured).

Reference Manual to LL API cross reference:

- CR1 AWD1CH LL_ADC_SetAnalogWDMonitChannels
- CR1 AWD1SGL LL_ADC_SetAnalogWDMonitChannels
- CR1 AWD1EN LL_ADC_SetAnalogWDMonitChannels

LL_ADC_GetAnalogWDMonitChannels

Function name

__STATIC_INLINE uint32_t LL_ADC_GetAnalogWDMonitChannels (ADC_TypeDef * ADCx)

Function description

Get ADC analog watchdog monitored channel.

Parameters

- **ADCx:** ADC instance

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_AWD_DISABLE
  - LL_ADC_AWD_ALL_CHANNELS_REG
  - LL_ADC_AWD_ALL_CHANNELS_INJ
  - LL_ADC_AWD_ALL_CHANNELS_REG_INJ
  - LL_ADC_AWD_CHANNEL_0_REG
  - LL_ADC_AWD_CHANNEL_0_INJ
  - LL_ADC_AWD_CHANNEL_0_REG_INJ
  - LL_ADC_AWD_CHANNEL_1_REG
  - LL_ADC_AWD_CHANNEL_1_INJ
  - LL_ADC_AWD_CHANNEL_1_REG_INJ
  - LL_ADC_AWD_CHANNEL_2_REG
  - LL_ADC_AWD_CHANNEL_2_INJ
  - LL_ADC_AWD_CHANNEL_2_REG_INJ
  - LL_ADC_AWD_CHANNEL_3_REG
  - LL_ADC_AWD_CHANNEL_3_INJ
  - LL_ADC_AWD_CHANNEL_3_REG_INJ
  - LL_ADC_AWD_CHANNEL_4_REG
  - LL_ADC_AWD_CHANNEL_4_INJ
  - LL_ADC_AWD_CHANNEL_4_REG_INJ
  - LL_ADC_AWD_CHANNEL_5_REG
  - LL_ADC_AWD_CHANNEL_5_INJ
  - LL_ADC_AWD_CHANNEL_5_REG_INJ
  - LL_ADC_AWD_CHANNEL_6_REG
  - LL_ADC_AWD_CHANNEL_6_INJ
  - LL_ADC_AWD_CHANNEL_6_REG_INJ
  - LL_ADC_AWD_CHANNEL_7_REG
  - LL_ADC_AWD_CHANNEL_7_INJ
  - LL_ADC_AWD_CHANNEL_7_REG_INJ
  - LL_ADC_AWD_CHANNEL_8_REG
  - LL_ADC_AWD_CHANNEL_8_INJ
  - LL_ADC_AWD_CHANNEL_8_REG_INJ
  - LL_ADC_AWD_CHANNEL_9_REG
  - LL_ADC_AWD_CHANNEL_9_INJ
  - LL_ADC_AWD_CHANNEL_9_REG_INJ
  - LL_ADC_AWD_CHANNEL_10_REG
  - LL_ADC_AWD_CHANNEL_10_INJ
  - LL_ADC_AWD_CHANNEL_10_REG_INJ
  - LL_ADC_AWD_CHANNEL_11_REG
  - LL_ADC_AWD_CHANNEL_11_INJ
  - LL_ADC_AWD_CHANNEL_11_REG_INJ
  - LL_ADC_AWD_CHANNEL_12_REG
  - LL_ADC_AWD_CHANNEL_12_INJ
  - LL_ADC_AWD_CHANNEL_12_REG_INJ
  - LL_ADC_AWD_CHANNEL_13_REG
  - LL_ADC_AWD_CHANNEL_13_INJ
  - LL_ADC_AWD_CHANNEL_13_REG_INJ
  - LL_ADC_AWD_CHANNEL_14_REG
  - LL_ADC_AWD_CHANNEL_14_INJ
  - LL_ADC_AWD_CHANNEL_14_REG_INJ
  - LL_ADC_AWD_CHANNEL_15_REG
  - LL_ADC_AWD_CHANNEL_15_INJ

Notes

- Usage of the returned channel number: To reinject this channel into another function LL_ADC_xxx: the returned channel number is only partly formatted on definition of literals LL_ADC_CHANNEL_x. Therefore, it has to be compared with parts of literals LL_ADC_CHANNEL_x or using helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB(). Then the selected literal LL_ADC_CHANNEL_x can be used as parameter for another function.To get the channel number in decimal format: process the returned value with the helper macro __LL_ADC_CHANNEL_TO_DECIMAL_NB(). Applicable only when the analog watchdog is set to monitor one channel.
- On this STM32 serie, there is only 1 kind of analog watchdog instance: AWD standard (instance AWD1): channels monitored: can monitor 1 channel or all channels.groups monitored: ADC groups regular and-or injected.resolution: resolution is not limited (corresponds to ADC resolution configured).

Reference Manual to LL API cross reference:

- CR1 AWD1CH LL_ADC_GetAnalogWDMonitChannels
- CR1 AWD1SGL LL_ADC_GetAnalogWDMonitChannels
- CR1 AWD1EN LL_ADC_GetAnalogWDMonitChannels

LL_ADC_SetAnalogWDThresholds

Function name

__STATIC_INLINE void LL_ADC_SetAnalogWDThresholds (ADC_TypeDef * ADCx, uint32_t AWDThresholdsHighLow, uint32_t AWDThresholdValue)

Function description

Set ADC analog watchdog threshold value of threshold high or low.

Parameters

- **ADCx:** ADC instance
- **AWDThresholdsHighLow:** This parameter can be one of the following values:
  - LL_ADC_AWD_THRESHOLD_HIGH
  - LL_ADC_AWD_THRESHOLD_LOW
- **AWDThresholdValue:** Value between Min_Data=0x000 and Max_Data=0xFFF

Return values

- **None:**

Notes

- In case of ADC resolution different of 12 bits, analog watchdog thresholds data require a specific shift. Use helper macro __LL_ADC_ANALOGWD_SET_THRESHOLD_RESOLUTION().
- On this STM32 serie, there is only 1 kind of analog watchdog instance: AWD standard (instance AWD1): channels monitored: can monitor 1 channel or all channels.groups monitored: ADC groups regular and-or injected.resolution: resolution is not limited (corresponds to ADC resolution configured).

Reference Manual to LL API cross reference:

- HTR HT LL_ADC_SetAnalogWDThresholds
- LTR LT LL_ADC_SetAnalogWDThresholds

LL_ADC_GetAnalogWDThresholds

Function name

__STATIC_INLINE uint32_t LL_ADC_GetAnalogWDThresholds (ADC_TypeDef * ADCx, uint32_t AWDThresholdsHighLow)

Function description

Get ADC analog watchdog threshold value of threshold high or threshold low.

Parameters

- **ADCx:** ADC instance
- **AWDThresholdsHighLow:** This parameter can be one of the following values:
  - LL_ADC_AWD_THRESHOLD_HIGH
  - LL_ADC_AWD_THRESHOLD_LOW

Return values

- **Value:** between Min_Data=0x000 and Max_Data=0xFFF

Notes

- In case of ADC resolution different of 12 bits, analog watchdog thresholds data require a specific shift. Use helper macro __LL_ADC_ANALOGWD_GET_THRESHOLD_RESOLUTION().

Reference Manual to LL API cross reference:

- HTR HT LL_ADC_GetAnalogWDThresholds
- LTR LT LL_ADC_GetAnalogWDThresholds

LL_ADC_SetMultimode

Function name

__STATIC_INLINE void LL_ADC_SetMultimode (ADC_Common_TypeDef * ADCxy_COMMON, uint32_t Multimode)

Function description

Set ADC multimode configuration to operate in independent mode or multimode (for devices with several ADC instances).

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())
- **Multimode:** This parameter can be one of the following values:
  - LL_ADC_MULTI_INDEPENDENT
  - LL_ADC_MULTI_DUAL_REG_SIMULT
  - LL_ADC_MULTI_DUAL_REG_INTERL
  - LL_ADC_MULTI_DUAL_INJ_SIMULT
  - LL_ADC_MULTI_DUAL_INJ_ALTERN
  - LL_ADC_MULTI_DUAL_REG_SIM_INJ_SIM
  - LL_ADC_MULTI_DUAL_REG_SIM_INJ_ALT
  - LL_ADC_MULTI_DUAL_REG_INT_INJ_SIM
  - LL_ADC_MULTI_TRIPLE_REG_SIM_INJ_SIM
  - LL_ADC_MULTI_TRIPLE_REG_SIM_INJ_ALT
  - LL_ADC_MULTI_TRIPLE_INJ_SIMULT
  - LL_ADC_MULTI_TRIPLE_REG_SIMULT
  - LL_ADC_MULTI_TRIPLE_REG_INTERL
  - LL_ADC_MULTI_TRIPLE_INJ_ALTERN

Return values

- **None:**

Notes

- If multimode configuration: the selected ADC instance is either master or slave depending on hardware. Refer to reference manual.

Reference Manual to LL API cross reference:

- CCR MULTI LL_ADC_SetMultimode

LL_ADC_GetMultimode

Function name

__STATIC_INLINE uint32_t LL_ADC_GetMultimode (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get ADC multimode configuration to operate in independent mode or multimode (for devices with several ADC instances).

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_MULTI_INDEPENDENT
  - LL_ADC_MULTI_DUAL_REG_SIMULT
  - LL_ADC_MULTI_DUAL_REG_INTERL
  - LL_ADC_MULTI_DUAL_INJ_SIMULT
  - LL_ADC_MULTI_DUAL_INJ_ALTERN
  - LL_ADC_MULTI_DUAL_REG_SIM_INJ_SIM
  - LL_ADC_MULTI_DUAL_REG_SIM_INJ_ALT
  - LL_ADC_MULTI_DUAL_REG_INT_INJ_SIM
  - LL_ADC_MULTI_TRIPLE_REG_SIM_INJ_SIM
  - LL_ADC_MULTI_TRIPLE_REG_SIM_INJ_ALT
  - LL_ADC_MULTI_TRIPLE_INJ_SIMULT
  - LL_ADC_MULTI_TRIPLE_REG_SIMULT
  - LL_ADC_MULTI_TRIPLE_REG_INTERL
  - LL_ADC_MULTI_TRIPLE_INJ_ALTERN

Notes

- If multimode configuration: the selected ADC instance is either master or slave depending on hardware. Refer to reference manual.

Reference Manual to LL API cross reference:

- CCR MULTI LL_ADC_GetMultimode

LL_ADC_SetMultiDMATransfer

Function name

__STATIC_INLINE void LL_ADC_SetMultiDMATransfer (ADC_Common_TypeDef * ADCxy_COMMON, uint32_t MultiDMATransfer)

Function description

Set ADC multimode conversion data transfer: no transfer or transfer by DMA.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())
- **MultiDMATransfer:** This parameter can be one of the following values:
  - LL_ADC_MULTI_REG_DMA_EACH_ADC
  - LL_ADC_MULTI_REG_DMA_LIMIT_1
  - LL_ADC_MULTI_REG_DMA_LIMIT_2
  - LL_ADC_MULTI_REG_DMA_LIMIT_3
  - LL_ADC_MULTI_REG_DMA_UNLMT_1
  - LL_ADC_MULTI_REG_DMA_UNLMT_2
  - LL_ADC_MULTI_REG_DMA_UNLMT_3

Return values

- **None:**

Notes

- If ADC multimode transfer by DMA is not selected: each ADC uses its own DMA channel, with its individual DMA transfer settings. If ADC multimode transfer by DMA is selected: One DMA channel is used for both ADC (DMA of ADC master) Specifies the DMA requests mode: Limited mode (One shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular.Unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transfers (number of ADC conversions). This ADC mode is intended to be used with DMA mode circular.
- If ADC DMA requests mode is set to unlimited and DMA is set to mode non-circular: when DMA transfers size will be reached, DMA will stop transfers of ADC conversions data ADC will raise an overrun error (overrun flag and interruption if enabled).
- How to retrieve multimode conversion data: Whatever multimode transfer by DMA setting: using function LL_ADC_REG_ReadMultiConversionData32(). If ADC multimode transfer by DMA is selected: conversion data is a raw data with ADC master and slave concatenated. A macro is available to get the conversion data of ADC master or ADC slave: see helper macro __LL_ADC_MULTI_CONV_DATA_MASTER_SLAVE().

Reference Manual to LL API cross reference:

- CCR MDMA LL_ADC_SetMultiDMATransfer
- CCR DDS LL_ADC_SetMultiDMATransfer

LL_ADC_GetMultiDMATransfer

Function name

__STATIC_INLINE uint32_t LL_ADC_GetMultiDMATransfer (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get ADC multimode conversion data transfer: no transfer or transfer by DMA.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_MULTI_REG_DMA_EACH_ADC
  - LL_ADC_MULTI_REG_DMA_LIMIT_1
  - LL_ADC_MULTI_REG_DMA_LIMIT_2
  - LL_ADC_MULTI_REG_DMA_LIMIT_3
  - LL_ADC_MULTI_REG_DMA_UNLMT_1
  - LL_ADC_MULTI_REG_DMA_UNLMT_2
  - LL_ADC_MULTI_REG_DMA_UNLMT_3

Notes

- If ADC multimode transfer by DMA is not selected: each ADC uses its own DMA channel, with its individual DMA transfer settings. If ADC multimode transfer by DMA is selected: One DMA channel is used for both ADC (DMA of ADC master) Specifies the DMA requests mode: Limited mode (One shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular.Unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transfers (number of ADC conversions). This ADC mode is intended to be used with DMA mode circular.
- If ADC DMA requests mode is set to unlimited and DMA is set to mode non-circular: when DMA transfers size will be reached, DMA will stop transfers of ADC conversions data ADC will raise an overrun error (overrun flag and interruption if enabled).
- How to retrieve multimode conversion data: Whatever multimode transfer by DMA setting: using function LL_ADC_REG_ReadMultiConversionData32(). If ADC multimode transfer by DMA is selected: conversion data is a raw data with ADC master and slave concatenated. A macro is available to get the conversion data of ADC master or ADC slave: see helper macro __LL_ADC_MULTI_CONV_DATA_MASTER_SLAVE().

Reference Manual to LL API cross reference:

- CCR MDMA LL_ADC_GetMultiDMATransfer
- CCR DDS LL_ADC_GetMultiDMATransfer

LL_ADC_SetMultiTwoSamplingDelay

Function name

__STATIC_INLINE void LL_ADC_SetMultiTwoSamplingDelay (ADC_Common_TypeDef * ADCxy_COMMON, uint32_t MultiTwoSamplingDelay)

Function description

Set ADC multimode delay between 2 sampling phases.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())
- **MultiTwoSamplingDelay:** This parameter can be one of the following values:
  - LL_ADC_MULTI_TWOSMP_DELAY_5CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_6CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_7CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_8CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_9CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_10CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_11CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_12CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_13CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_14CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_15CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_16CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_17CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_18CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_19CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_20CYCLES

Return values

- **None:**

Notes

- The sampling delay range depends on ADC resolution: ADC resolution 12 bits can have maximum delay of 12 cycles.ADC resolution 10 bits can have maximum delay of 10 cycles.ADC resolution 8 bits can have maximum delay of 8 cycles.ADC resolution 6 bits can have maximum delay of 6 cycles.

Reference Manual to LL API cross reference:

- CCR DELAY LL_ADC_SetMultiTwoSamplingDelay

LL_ADC_GetMultiTwoSamplingDelay

Function name

__STATIC_INLINE uint32_t LL_ADC_GetMultiTwoSamplingDelay (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get ADC multimode delay between 2 sampling phases.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **Returned:** value can be one of the following values:
  - LL_ADC_MULTI_TWOSMP_DELAY_5CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_6CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_7CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_8CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_9CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_10CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_11CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_12CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_13CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_14CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_15CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_16CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_17CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_18CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_19CYCLES
  - LL_ADC_MULTI_TWOSMP_DELAY_20CYCLES

Reference Manual to LL API cross reference:

- CCR DELAY LL_ADC_GetMultiTwoSamplingDelay

LL_ADC_Enable

Function name

__STATIC_INLINE void LL_ADC_Enable (ADC_TypeDef * ADCx)

Function description

Enable the selected ADC instance.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Notes

- On this STM32 serie, after ADC enable, a delay for ADC internal analog stabilization is required before performing a ADC conversion start. Refer to device datasheet, parameter tSTAB.

Reference Manual to LL API cross reference:

- CR2 ADON LL_ADC_Enable

LL_ADC_Disable

Function name

__STATIC_INLINE void LL_ADC_Disable (ADC_TypeDef * ADCx)

Function description

Disable the selected ADC instance.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 ADON LL_ADC_Disable

LL_ADC_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_ADC_IsEnabled (ADC_TypeDef * ADCx)

Function description

Get the selected ADC instance enable state.

Parameters

- **ADCx:** ADC instance

Return values

- **0:** ADC is disabled, 1: ADC is enabled.

Reference Manual to LL API cross reference:

- CR2 ADON LL_ADC_IsEnabled

LL_ADC_REG_StartConversionSWStart

Function name

__STATIC_INLINE void LL_ADC_REG_StartConversionSWStart (ADC_TypeDef * ADCx)

Function description

Start ADC group regular conversion.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Notes

- On this STM32 serie, this function is relevant only for internal trigger (SW start), not for external trigger: If ADC trigger has been set to software start, ADC conversion starts immediately.If ADC trigger has been set to external trigger, ADC conversion start must be performed using function LL_ADC_REG_StartConversionExtTrig(). (if external trigger edge would have been set during ADC other settings, ADC conversion would start at trigger event as soon as ADC is enabled).

Reference Manual to LL API cross reference:

- CR2 SWSTART LL_ADC_REG_StartConversionSWStart

LL_ADC_REG_StartConversionExtTrig

Function name

__STATIC_INLINE void LL_ADC_REG_StartConversionExtTrig (ADC_TypeDef * ADCx, uint32_t ExternalTriggerEdge)

Function description

Start ADC group regular conversion from external trigger.

Parameters

- **ExternalTriggerEdge:** This parameter can be one of the following values:
  - LL_ADC_REG_TRIG_EXT_RISING
  - LL_ADC_REG_TRIG_EXT_FALLING
  - LL_ADC_REG_TRIG_EXT_RISINGFALLING
- **ADCx:** ADC instance

Return values

- **None:**

Notes

- ADC conversion will start at next trigger event (on the selected trigger edge) following the ADC start conversion command.
- On this STM32 serie, this function is relevant for ADC conversion start from external trigger. If internal trigger (SW start) is needed, perform ADC conversion start using function LL_ADC_REG_StartConversionSWStart().

Reference Manual to LL API cross reference:

- CR2 EXTEN LL_ADC_REG_StartConversionExtTrig

LL_ADC_REG_StopConversionExtTrig

Function name

__STATIC_INLINE void LL_ADC_REG_StopConversionExtTrig (ADC_TypeDef * ADCx)

Function description

Stop ADC group regular conversion from external trigger.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Notes

- No more ADC conversion will start at next trigger event following the ADC stop conversion command. If a conversion is on-going, it will be completed.
- On this STM32 serie, there is no specific command to stop a conversion on-going or to stop ADC converting in continuous mode. These actions can be performed using function LL_ADC_Disable().

Reference Manual to LL API cross reference:

- CR2 EXTEN LL_ADC_REG_StopConversionExtTrig

LL_ADC_REG_ReadConversionData32

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_ReadConversionData32 (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion data, range fit for all ADC configurations: all ADC resolutions and all oversampling increased data width (for devices with feature oversampling).

Parameters

- **ADCx:** ADC instance

Return values

- **Value:** between Min_Data=0x00000000 and Max_Data=0xFFFFFFFF

Reference Manual to LL API cross reference:

- DR RDATA LL_ADC_REG_ReadConversionData32

LL_ADC_REG_ReadConversionData12

Function name

__STATIC_INLINE uint16_t LL_ADC_REG_ReadConversionData12 (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion data, range fit for ADC resolution 12 bits.

Parameters

- **ADCx:** ADC instance

Return values

- **Value:** between Min_Data=0x000 and Max_Data=0xFFF

Notes

- For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_REG_ReadConversionData32.

Reference Manual to LL API cross reference:

- DR RDATA LL_ADC_REG_ReadConversionData12

LL_ADC_REG_ReadConversionData10

Function name

__STATIC_INLINE uint16_t LL_ADC_REG_ReadConversionData10 (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion data, range fit for ADC resolution 10 bits.

Parameters

- **ADCx:** ADC instance

Return values

- **Value:** between Min_Data=0x000 and Max_Data=0x3FF

Notes

- For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_REG_ReadConversionData32.

Reference Manual to LL API cross reference:

- DR RDATA LL_ADC_REG_ReadConversionData10

LL_ADC_REG_ReadConversionData8

Function name

__STATIC_INLINE uint8_t LL_ADC_REG_ReadConversionData8 (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion data, range fit for ADC resolution 8 bits.

Parameters

- **ADCx:** ADC instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Notes

- For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_REG_ReadConversionData32.

Reference Manual to LL API cross reference:

- DR RDATA LL_ADC_REG_ReadConversionData8

LL_ADC_REG_ReadConversionData6

Function name

__STATIC_INLINE uint8_t LL_ADC_REG_ReadConversionData6 (ADC_TypeDef * ADCx)

Function description

Get ADC group regular conversion data, range fit for ADC resolution 6 bits.

Parameters

- **ADCx:** ADC instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x3F

Notes

- For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_REG_ReadConversionData32.

Reference Manual to LL API cross reference:

- DR RDATA LL_ADC_REG_ReadConversionData6

LL_ADC_REG_ReadMultiConversionData32

Function name

__STATIC_INLINE uint32_t LL_ADC_REG_ReadMultiConversionData32 (ADC_Common_TypeDef * ADCxy_COMMON, uint32_t ConversionData)

Function description

Get ADC multimode conversion data of ADC master, ADC slave or raw data with ADC master and slave concatenated.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())
- **ConversionData:** This parameter can be one of the following values:
  - LL_ADC_MULTI_MASTER
  - LL_ADC_MULTI_SLAVE
  - LL_ADC_MULTI_MASTER_SLAVE

Return values

- **Value:** between Min_Data=0x00000000 and Max_Data=0xFFFFFFFF

Notes

- If raw data with ADC master and slave concatenated is retrieved, a macro is available to get the conversion data of ADC master or ADC slave: see helper macro __LL_ADC_MULTI_CONV_DATA_MASTER_SLAVE(). (however this macro is mainly intended for multimode transfer by DMA, because this function can do the same by getting multimode conversion data of ADC master or ADC slave separately).

Reference Manual to LL API cross reference:

- CDR DATA1 LL_ADC_REG_ReadMultiConversionData32
- CDR DATA2 LL_ADC_REG_ReadMultiConversionData32

LL_ADC_INJ_StartConversionSWStart

Function name

__STATIC_INLINE void LL_ADC_INJ_StartConversionSWStart (ADC_TypeDef * ADCx)

Function description

Start ADC group injected conversion.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Notes

- On this STM32 serie, this function is relevant only for internal trigger (SW start), not for external trigger: If ADC trigger has been set to software start, ADC conversion starts immediately.If ADC trigger has been set to external trigger, ADC conversion start must be performed using function LL_ADC_INJ_StartConversionExtTrig(). (if external trigger edge would have been set during ADC other settings, ADC conversion would start at trigger event as soon as ADC is enabled).

Reference Manual to LL API cross reference:

- CR2 JSWSTART LL_ADC_INJ_StartConversionSWStart

LL_ADC_INJ_StartConversionExtTrig

Function name

__STATIC_INLINE void LL_ADC_INJ_StartConversionExtTrig (ADC_TypeDef * ADCx, uint32_t ExternalTriggerEdge)

Function description

Start ADC group injected conversion from external trigger.

Parameters

- **ExternalTriggerEdge:** This parameter can be one of the following values:
  - LL_ADC_INJ_TRIG_EXT_RISING
  - LL_ADC_INJ_TRIG_EXT_FALLING
  - LL_ADC_INJ_TRIG_EXT_RISINGFALLING
- **ADCx:** ADC instance

Return values

- **None:**

Notes

- ADC conversion will start at next trigger event (on the selected trigger edge) following the ADC start conversion command.
- On this STM32 serie, this function is relevant for ADC conversion start from external trigger. If internal trigger (SW start) is needed, perform ADC conversion start using function LL_ADC_INJ_StartConversionSWStart().

Reference Manual to LL API cross reference:

- CR2 JEXTEN LL_ADC_INJ_StartConversionExtTrig

LL_ADC_INJ_StopConversionExtTrig

Function name

__STATIC_INLINE void LL_ADC_INJ_StopConversionExtTrig (ADC_TypeDef * ADCx)

Function description

Stop ADC group injected conversion from external trigger.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Notes

- No more ADC conversion will start at next trigger event following the ADC stop conversion command. If a conversion is on-going, it will be completed.
- On this STM32 serie, there is no specific command to stop a conversion on-going or to stop ADC converting in continuous mode. These actions can be performed using function LL_ADC_Disable().

Reference Manual to LL API cross reference:

- CR2 JEXTEN LL_ADC_INJ_StopConversionExtTrig

LL_ADC_INJ_ReadConversionData32

Function name

__STATIC_INLINE uint32_t LL_ADC_INJ_ReadConversionData32 (ADC_TypeDef * ADCx, uint32_t Rank)

Function description

Get ADC group regular conversion data, range fit for all ADC configurations: all ADC resolutions and all oversampling increased data width (for devices with feature oversampling).

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4

Return values

- **Value:** between Min_Data=0x00000000 and Max_Data=0xFFFFFFFF

Reference Manual to LL API cross reference:

- JDR1 JDATA LL_ADC_INJ_ReadConversionData32
- JDR2 JDATA LL_ADC_INJ_ReadConversionData32
- JDR3 JDATA LL_ADC_INJ_ReadConversionData32
- JDR4 JDATA LL_ADC_INJ_ReadConversionData32

LL_ADC_INJ_ReadConversionData12

Function name

__STATIC_INLINE uint16_t LL_ADC_INJ_ReadConversionData12 (ADC_TypeDef * ADCx, uint32_t Rank)

Function description

Get ADC group injected conversion data, range fit for ADC resolution 12 bits.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4

Return values

- **Value:** between Min_Data=0x000 and Max_Data=0xFFF

Notes

- For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_INJ_ReadConversionData32.

Reference Manual to LL API cross reference:

- JDR1 JDATA LL_ADC_INJ_ReadConversionData12
- JDR2 JDATA LL_ADC_INJ_ReadConversionData12
- JDR3 JDATA LL_ADC_INJ_ReadConversionData12
- JDR4 JDATA LL_ADC_INJ_ReadConversionData12

LL_ADC_INJ_ReadConversionData10

Function name

__STATIC_INLINE uint16_t LL_ADC_INJ_ReadConversionData10 (ADC_TypeDef * ADCx, uint32_t Rank)

Function description

Get ADC group injected conversion data, range fit for ADC resolution 10 bits.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4

Return values

- **Value:** between Min_Data=0x000 and Max_Data=0x3FF

Notes

- For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_INJ_ReadConversionData32.

Reference Manual to LL API cross reference:

- JDR1 JDATA LL_ADC_INJ_ReadConversionData10
- JDR2 JDATA LL_ADC_INJ_ReadConversionData10
- JDR3 JDATA LL_ADC_INJ_ReadConversionData10
- JDR4 JDATA LL_ADC_INJ_ReadConversionData10

LL_ADC_INJ_ReadConversionData8

Function name

__STATIC_INLINE uint8_t LL_ADC_INJ_ReadConversionData8 (ADC_TypeDef * ADCx, uint32_t Rank)

Function description

Get ADC group injected conversion data, range fit for ADC resolution 8 bits.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Notes

- For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_INJ_ReadConversionData32.

Reference Manual to LL API cross reference:

- JDR1 JDATA LL_ADC_INJ_ReadConversionData8
- JDR2 JDATA LL_ADC_INJ_ReadConversionData8
- JDR3 JDATA LL_ADC_INJ_ReadConversionData8
- JDR4 JDATA LL_ADC_INJ_ReadConversionData8

LL_ADC_INJ_ReadConversionData6

Function name

__STATIC_INLINE uint8_t LL_ADC_INJ_ReadConversionData6 (ADC_TypeDef * ADCx, uint32_t Rank)

Function description

Get ADC group injected conversion data, range fit for ADC resolution 6 bits.

Parameters

- **ADCx:** ADC instance
- **Rank:** This parameter can be one of the following values:
  - LL_ADC_INJ_RANK_1
  - LL_ADC_INJ_RANK_2
  - LL_ADC_INJ_RANK_3
  - LL_ADC_INJ_RANK_4

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x3F

Notes

- For devices with feature oversampling: Oversampling can increase data width, function for extended range may be needed: LL_ADC_INJ_ReadConversionData32.

Reference Manual to LL API cross reference:

- JDR1 JDATA LL_ADC_INJ_ReadConversionData6
- JDR2 JDATA LL_ADC_INJ_ReadConversionData6
- JDR3 JDATA LL_ADC_INJ_ReadConversionData6
- JDR4 JDATA LL_ADC_INJ_ReadConversionData6

LL_ADC_IsActiveFlag_EOCS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_EOCS (ADC_TypeDef * ADCx)

Function description

Get flag ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.

Parameters

- **ADCx:** ADC instance

Return values

- **State:** of bit (1 or 0).

Notes

- To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().

Reference Manual to LL API cross reference:

- SR EOC LL_ADC_IsActiveFlag_EOCS

LL_ADC_IsActiveFlag_OVR

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_OVR (ADC_TypeDef * ADCx)

Function description

Get flag ADC group regular overrun.

Parameters

- **ADCx:** ADC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR OVR LL_ADC_IsActiveFlag_OVR

LL_ADC_IsActiveFlag_JEOS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_JEOS (ADC_TypeDef * ADCx)

Function description

Get flag ADC group injected end of sequence conversions.

Parameters

- **ADCx:** ADC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR JEOC LL_ADC_IsActiveFlag_JEOS

LL_ADC_IsActiveFlag_AWD1

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_AWD1 (ADC_TypeDef * ADCx)

Function description

Get flag ADC analog watchdog 1 flag.

Parameters

- **ADCx:** ADC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR AWD LL_ADC_IsActiveFlag_AWD1

LL_ADC_ClearFlag_EOCS

Function name

__STATIC_INLINE void LL_ADC_ClearFlag_EOCS (ADC_TypeDef * ADCx)

Function description

Clear flag ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Notes

- To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().

Reference Manual to LL API cross reference:

- SR EOC LL_ADC_ClearFlag_EOCS

LL_ADC_ClearFlag_OVR

Function name

__STATIC_INLINE void LL_ADC_ClearFlag_OVR (ADC_TypeDef * ADCx)

Function description

Clear flag ADC group regular overrun.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR OVR LL_ADC_ClearFlag_OVR

LL_ADC_ClearFlag_JEOS

Function name

__STATIC_INLINE void LL_ADC_ClearFlag_JEOS (ADC_TypeDef * ADCx)

Function description

Clear flag ADC group injected end of sequence conversions.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR JEOC LL_ADC_ClearFlag_JEOS

LL_ADC_ClearFlag_AWD1

Function name

__STATIC_INLINE void LL_ADC_ClearFlag_AWD1 (ADC_TypeDef * ADCx)

Function description

Clear flag ADC analog watchdog 1.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR AWD LL_ADC_ClearFlag_AWD1

LL_ADC_IsActiveFlag_MST_EOCS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_MST_EOCS (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration, of the ADC master.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Notes

- To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().

Reference Manual to LL API cross reference:

- CSR EOC1 LL_ADC_IsActiveFlag_MST_EOCS

LL_ADC_IsActiveFlag_SLV1_EOCS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_SLV1_EOCS (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration, of the ADC slave 1.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Notes

- To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().

Reference Manual to LL API cross reference:

- CSR EOC2 LL_ADC_IsActiveFlag_SLV1_EOCS

LL_ADC_IsActiveFlag_SLV2_EOCS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_SLV2_EOCS (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration, of the ADC slave 2.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Notes

- To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().

Reference Manual to LL API cross reference:

- CSR EOC3 LL_ADC_IsActiveFlag_SLV2_EOCS

LL_ADC_IsActiveFlag_MST_OVR

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_MST_OVR (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group regular overrun of the ADC master.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR OVR1 LL_ADC_IsActiveFlag_MST_OVR

LL_ADC_IsActiveFlag_SLV1_OVR

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_SLV1_OVR (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group regular overrun of the ADC slave 1.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR OVR2 LL_ADC_IsActiveFlag_SLV1_OVR

LL_ADC_IsActiveFlag_SLV2_OVR

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_SLV2_OVR (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group regular overrun of the ADC slave 2.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR OVR3 LL_ADC_IsActiveFlag_SLV2_OVR

LL_ADC_IsActiveFlag_MST_JEOS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_MST_JEOS (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group injected end of sequence conversions of the ADC master.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR JEOC LL_ADC_IsActiveFlag_MST_EOCS

LL_ADC_IsActiveFlag_SLV1_JEOS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_SLV1_JEOS (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group injected end of sequence conversions of the ADC slave 1.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR JEOC2 LL_ADC_IsActiveFlag_SLV1_JEOS

LL_ADC_IsActiveFlag_SLV2_JEOS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_SLV2_JEOS (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC group injected end of sequence conversions of the ADC slave 2.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR JEOC3 LL_ADC_IsActiveFlag_SLV2_JEOS

LL_ADC_IsActiveFlag_MST_AWD1

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_MST_AWD1 (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode ADC analog watchdog 1 of the ADC master.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR AWD1 LL_ADC_IsActiveFlag_MST_AWD1

LL_ADC_IsActiveFlag_SLV1_AWD1

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_SLV1_AWD1 (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode analog watchdog 1 of the ADC slave 1.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR AWD2 LL_ADC_IsActiveFlag_SLV1_AWD1

LL_ADC_IsActiveFlag_SLV2_AWD1

Function name

__STATIC_INLINE uint32_t LL_ADC_IsActiveFlag_SLV2_AWD1 (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

Get flag multimode analog watchdog 1 of the ADC slave 2.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR AWD3 LL_ADC_IsActiveFlag_SLV2_AWD1

LL_ADC_EnableIT_EOCS

Function name

__STATIC_INLINE void LL_ADC_EnableIT_EOCS (ADC_TypeDef * ADCx)

Function description

Enable interruption ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Notes

- To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().

Reference Manual to LL API cross reference:

- CR1 EOCIE LL_ADC_EnableIT_EOCS

LL_ADC_EnableIT_OVR

Function name

__STATIC_INLINE void LL_ADC_EnableIT_OVR (ADC_TypeDef * ADCx)

Function description

Enable ADC group regular interruption overrun.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 OVRIE LL_ADC_EnableIT_OVR

LL_ADC_EnableIT_JEOS

Function name

__STATIC_INLINE void LL_ADC_EnableIT_JEOS (ADC_TypeDef * ADCx)

Function description

Enable interruption ADC group injected end of sequence conversions.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 JEOCIE LL_ADC_EnableIT_JEOS

LL_ADC_EnableIT_AWD1

Function name

__STATIC_INLINE void LL_ADC_EnableIT_AWD1 (ADC_TypeDef * ADCx)

Function description

Enable interruption ADC analog watchdog 1.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 AWDIE LL_ADC_EnableIT_AWD1

LL_ADC_DisableIT_EOCS

Function name

__STATIC_INLINE void LL_ADC_DisableIT_EOCS (ADC_TypeDef * ADCx)

Function description

Disable interruption ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Notes

- To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion().

Reference Manual to LL API cross reference:

- CR1 EOCIE LL_ADC_DisableIT_EOCS

LL_ADC_DisableIT_OVR

Function name

__STATIC_INLINE void LL_ADC_DisableIT_OVR (ADC_TypeDef * ADCx)

Function description

Disable interruption ADC group regular overrun.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 OVRIE LL_ADC_DisableIT_OVR

LL_ADC_DisableIT_JEOS

Function name

__STATIC_INLINE void LL_ADC_DisableIT_JEOS (ADC_TypeDef * ADCx)

Function description

Disable interruption ADC group injected end of sequence conversions.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 JEOCIE LL_ADC_EnableIT_JEOS

LL_ADC_DisableIT_AWD1

Function name

__STATIC_INLINE void LL_ADC_DisableIT_AWD1 (ADC_TypeDef * ADCx)

Function description

Disable interruption ADC analog watchdog 1.

Parameters

- **ADCx:** ADC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 AWDIE LL_ADC_EnableIT_AWD1

LL_ADC_IsEnabledIT_EOCS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_EOCS (ADC_TypeDef * ADCx)

Function description

Get state of interruption ADC group regular end of unitary conversion or end of sequence conversions, depending on ADC configuration.

Parameters

- **ADCx:** ADC instance

Return values

- **State:** of bit (1 or 0).

Notes

- To configure flag of end of conversion, use function LL_ADC_REG_SetFlagEndOfConversion(). (0: interrupt disabled, 1: interrupt enabled)

Reference Manual to LL API cross reference:

- CR1 EOCIE LL_ADC_IsEnabledIT_EOCS

LL_ADC_IsEnabledIT_OVR

Function name

__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_OVR (ADC_TypeDef * ADCx)

Function description

Get state of interruption ADC group regular overrun (0: interrupt disabled, 1: interrupt enabled).

Parameters

- **ADCx:** ADC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 OVRIE LL_ADC_IsEnabledIT_OVR

LL_ADC_IsEnabledIT_JEOS

Function name

__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_JEOS (ADC_TypeDef * ADCx)

Function description

Get state of interruption ADC group injected end of sequence conversions (0: interrupt disabled, 1: interrupt enabled).

Parameters

- **ADCx:** ADC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 JEOCIE LL_ADC_EnableIT_JEOS

LL_ADC_IsEnabledIT_AWD1

Function name

__STATIC_INLINE uint32_t LL_ADC_IsEnabledIT_AWD1 (ADC_TypeDef * ADCx)

Function description

Get state of interruption ADC analog watchdog 1 (0: interrupt disabled, 1: interrupt enabled).

Parameters

- **ADCx:** ADC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 AWDIE LL_ADC_EnableIT_AWD1

LL_ADC_CommonDeInit

Function name

ErrorStatus LL_ADC_CommonDeInit (ADC_Common_TypeDef * ADCxy_COMMON)

Function description

De-initialize registers of all ADC instances belonging to the same ADC common instance to their default reset values.

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: ADC common registers are de-initialized
  - ERROR: not applicable

LL_ADC_CommonInit

Function name

ErrorStatus LL_ADC_CommonInit (ADC_Common_TypeDef * ADCxy_COMMON, LL_ADC_CommonInitTypeDef * ADC_CommonInitStruct)

Function description

Initialize some features of ADC common parameters (all ADC instances belonging to the same ADC common instance) and multimode (for devices with several ADC instances available).

Parameters

- **ADCxy_COMMON:** ADC common instance (can be set directly from CMSIS definition or by using helper macro __LL_ADC_COMMON_INSTANCE())
- **ADC_CommonInitStruct:** Pointer to a LL_ADC_CommonInitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: ADC common registers are initialized
  - ERROR: ADC common registers are not initialized

Notes

- The setting of ADC common parameters is conditioned to ADC instances state: All ADC instances belonging to the same ADC common instance must be disabled.

LL_ADC_CommonStructInit

Function name

void LL_ADC_CommonStructInit (LL_ADC_CommonInitTypeDef * ADC_CommonInitStruct)

Function description

Set each LL_ADC_CommonInitTypeDef field to default value.

Parameters

- **ADC_CommonInitStruct:** Pointer to a LL_ADC_CommonInitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

LL_ADC_DeInit

Function name

ErrorStatus LL_ADC_DeInit (ADC_TypeDef * ADCx)

Function description

De-initialize registers of the selected ADC instance to their default reset values.

Parameters

- **ADCx:** ADC instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: ADC registers are de-initialized
  - ERROR: ADC registers are not de-initialized

Notes

- To reset all ADC instances quickly (perform a hard reset), use function LL_ADC_CommonDeInit().

LL_ADC_Init

Function name

ErrorStatus LL_ADC_Init (ADC_TypeDef * ADCx, LL_ADC_InitTypeDef * ADC_InitStruct)

Function description

Initialize some features of ADC instance.

Parameters

- **ADCx:** ADC instance
- **ADC_InitStruct:** Pointer to a LL_ADC_REG_InitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: ADC registers are initialized
  - ERROR: ADC registers are not initialized

Notes

- These parameters have an impact on ADC scope: ADC instance. Affects both group regular and group injected (availability of ADC group injected depends on STM32 families). Refer to corresponding unitary functions into Configuration of ADC hierarchical scope: ADC instance .
- The setting of these parameters by function LL_ADC_Init() is conditioned to ADC state: ADC instance must be disabled. This condition is applied to all ADC features, for efficiency and compatibility over all STM32 families. However, the different features can be set under different ADC state conditions (setting possible with ADC enabled without conversion on going, ADC enabled with conversion on going, ...) Each feature can be updated afterwards with a unitary function and potentially with ADC in a different state than disabled, refer to description of each function for setting conditioned to ADC state.
- After using this function, some other features must be configured using LL unitary functions. The minimum configuration remaining to be done is: Set ADC group regular or group injected sequencer: map channel on the selected sequencer rank. Refer to function LL_ADC_REG_SetSequencerRanks().Set ADC channel sampling time Refer to function LL_ADC_SetChannelSamplingTime();

LL_ADC_StructInit

Function name

void LL_ADC_StructInit (LL_ADC_InitTypeDef * ADC_InitStruct)

Function description

Set each LL_ADC_InitTypeDef field to default value.

Parameters

- **ADC_InitStruct:** Pointer to a LL_ADC_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

LL_ADC_REG_Init

Function name

ErrorStatus LL_ADC_REG_Init (ADC_TypeDef * ADCx, LL_ADC_REG_InitTypeDef * ADC_REG_InitStruct)

Function description

Initialize some features of ADC group regular.

Parameters

- **ADCx:** ADC instance
- **ADC_REG_InitStruct:** Pointer to a LL_ADC_REG_InitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: ADC registers are initialized
  - ERROR: ADC registers are not initialized

Notes

- These parameters have an impact on ADC scope: ADC group regular. Refer to corresponding unitary functions into Configuration of ADC hierarchical scope: group regular (functions with prefix "REG").
- The setting of these parameters by function LL_ADC_Init() is conditioned to ADC state: ADC instance must be disabled. This condition is applied to all ADC features, for efficiency and compatibility over all STM32 families. However, the different features can be set under different ADC state conditions (setting possible with ADC enabled without conversion on going, ADC enabled with conversion on going, ...) Each feature can be updated afterwards with a unitary function and potentially with ADC in a different state than disabled, refer to description of each function for setting conditioned to ADC state.
- After using this function, other features must be configured using LL unitary functions. The minimum configuration remaining to be done is: Set ADC group regular or group injected sequencer: map channel on the selected sequencer rank. Refer to function LL_ADC_REG_SetSequencerRanks().Set ADC channel sampling time Refer to function LL_ADC_SetChannelSamplingTime();

LL_ADC_REG_StructInit

Function name

void LL_ADC_REG_StructInit (LL_ADC_REG_InitTypeDef * ADC_REG_InitStruct)

Function description

Set each LL_ADC_REG_InitTypeDef field to default value.

Parameters

- **ADC_REG_InitStruct:** Pointer to a LL_ADC_REG_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

LL_ADC_INJ_Init

Function name

ErrorStatus LL_ADC_INJ_Init (ADC_TypeDef * ADCx, LL_ADC_INJ_InitTypeDef * ADC_INJ_InitStruct)

Function description

Initialize some features of ADC group injected.

Parameters

- **ADCx:** ADC instance
- **ADC_INJ_InitStruct:** Pointer to a LL_ADC_INJ_InitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: ADC registers are initialized
  - ERROR: ADC registers are not initialized

Notes

- These parameters have an impact on ADC scope: ADC group injected. Refer to corresponding unitary functions into Configuration of ADC hierarchical scope: group regular (functions with prefix "INJ").
- The setting of these parameters by function LL_ADC_Init() is conditioned to ADC state: ADC instance must be disabled. This condition is applied to all ADC features, for efficiency and compatibility over all STM32 families. However, the different features can be set under different ADC state conditions (setting possible with ADC enabled without conversion on going, ADC enabled with conversion on going, ...) Each feature can be updated afterwards with a unitary function and potentially with ADC in a different state than disabled, refer to description of each function for setting conditioned to ADC state.
- After using this function, other features must be configured using LL unitary functions. The minimum configuration remaining to be done is: Set ADC group injected sequencer: map channel on the selected sequencer rank. Refer to function LL_ADC_INJ_SetSequencerRanks().Set ADC channel sampling time Refer to function LL_ADC_SetChannelSamplingTime();

LL_ADC_INJ_StructInit

Function name

void LL_ADC_INJ_StructInit (LL_ADC_INJ_InitTypeDef * ADC_INJ_InitStruct)

Function description

Set each LL_ADC_INJ_InitTypeDef field to default value.

Parameters

- **ADC_INJ_InitStruct:** Pointer to a LL_ADC_INJ_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

73.3 ADC Firmware driver defines

The following section lists the various define and macros of the module.

73.3.1 ADC

ADC

Analog watchdog - Monitored channels

LL_ADC_AWD_DISABLE

ADC analog watchdog monitoring disabled

LL_ADC_AWD_ALL_CHANNELS_REG

ADC analog watchdog monitoring of all channels, converted by group regular only

LL_ADC_AWD_ALL_CHANNELS_INJ

ADC analog watchdog monitoring of all channels, converted by group injected only

LL_ADC_AWD_ALL_CHANNELS_REG_INJ

ADC analog watchdog monitoring of all channels, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_0_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN0, converted by group regular only

LL_ADC_AWD_CHANNEL_0_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN0, converted by group injected only

LL_ADC_AWD_CHANNEL_0_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN0, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_1_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN1, converted by group regular only

LL_ADC_AWD_CHANNEL_1_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN1, converted by group injected only

LL_ADC_AWD_CHANNEL_1_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN1, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_2_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN2, converted by group regular only

LL_ADC_AWD_CHANNEL_2_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN2, converted by group injected only

LL_ADC_AWD_CHANNEL_2_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN2, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_3_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN3, converted by group regular only

LL_ADC_AWD_CHANNEL_3_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN3, converted by group injected only

LL_ADC_AWD_CHANNEL_3_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN3, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_4_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN4, converted by group regular only

LL_ADC_AWD_CHANNEL_4_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN4, converted by group injected only

LL_ADC_AWD_CHANNEL_4_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN4, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_5_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN5, converted by group regular only

LL_ADC_AWD_CHANNEL_5_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN5, converted by group injected only

LL_ADC_AWD_CHANNEL_5_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN5, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_6_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN6, converted by group regular only

LL_ADC_AWD_CHANNEL_6_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN6, converted by group injected only

LL_ADC_AWD_CHANNEL_6_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN6, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_7_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN7, converted by group regular only

LL_ADC_AWD_CHANNEL_7_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN7, converted by group injected only

LL_ADC_AWD_CHANNEL_7_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN7, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_8_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN8, converted by group regular only

LL_ADC_AWD_CHANNEL_8_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN8, converted by group injected only

LL_ADC_AWD_CHANNEL_8_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN8, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_9_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN9, converted by group regular only

LL_ADC_AWD_CHANNEL_9_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN9, converted by group injected only

LL_ADC_AWD_CHANNEL_9_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN9, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_10_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN10, converted by group regular only

LL_ADC_AWD_CHANNEL_10_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN10, converted by group injected only

LL_ADC_AWD_CHANNEL_10_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN10, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_11_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN11, converted by group regular only

LL_ADC_AWD_CHANNEL_11_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN11, converted by group injected only

LL_ADC_AWD_CHANNEL_11_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN11, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_12_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN12, converted by group regular only

LL_ADC_AWD_CHANNEL_12_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN12, converted by group injected only

LL_ADC_AWD_CHANNEL_12_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN12, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_13_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN13, converted by group regular only

LL_ADC_AWD_CHANNEL_13_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN13, converted by group injected only

LL_ADC_AWD_CHANNEL_13_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN13, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_14_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN14, converted by group regular only

LL_ADC_AWD_CHANNEL_14_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN14, converted by group injected only

LL_ADC_AWD_CHANNEL_14_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN14, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_15_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN15, converted by group regular only

LL_ADC_AWD_CHANNEL_15_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN15, converted by group injected only

LL_ADC_AWD_CHANNEL_15_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN15, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_16_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN16, converted by group regular only

LL_ADC_AWD_CHANNEL_16_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN16, converted by group injected only

LL_ADC_AWD_CHANNEL_16_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN16, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_17_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN17, converted by group regular only

LL_ADC_AWD_CHANNEL_17_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN17, converted by group injected only

LL_ADC_AWD_CHANNEL_17_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN17, converted by either group regular or injected

LL_ADC_AWD_CHANNEL_18_REG

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN18, converted by group regular only

LL_ADC_AWD_CHANNEL_18_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN18, converted by group injected only

LL_ADC_AWD_CHANNEL_18_REG_INJ

ADC analog watchdog monitoring of ADC external channel (channel connected to GPIO pin) ADCx_IN18, converted by either group regular or injected

LL_ADC_AWD_CH_VREFINT_REG

ADC analog watchdog monitoring of ADC internal channel connected to VrefInt: Internal voltage reference, converted by group regular only

LL_ADC_AWD_CH_VREFINT_INJ

ADC analog watchdog monitoring of ADC internal channel connected to VrefInt: Internal voltage reference, converted by group injected only

LL_ADC_AWD_CH_VREFINT_REG_INJ

ADC analog watchdog monitoring of ADC internal channel connected to VrefInt: Internal voltage reference, converted by either group regular or injected

LL_ADC_AWD_CH_VBAT_REG

ADC analog watchdog monitoring of ADC internal channel connected to Vbat/3: Vbat voltage through a divider ladder of factor 1/3 to have Vbat always below Vdda, converted by group regular only

LL_ADC_AWD_CH_VBAT_INJ

ADC analog watchdog monitoring of ADC internal channel connected to Vbat/3: Vbat voltage through a divider ladder of factor 1/3 to have Vbat always below Vdda, converted by group injected only

LL_ADC_AWD_CH_VBAT_REG_INJ

ADC analog watchdog monitoring of ADC internal channel connected to Vbat/3: Vbat voltage through a divider ladder of factor 1/3 to have Vbat always below Vdda

LL_ADC_AWD_CH_TEMPSENSOR_REG

ADC analog watchdog monitoring of ADC internal channel connected to Temperature sensor, converted by group regular only. This internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.

LL_ADC_AWD_CH_TEMPSENSOR_INJ

ADC analog watchdog monitoring of ADC internal channel connected to Temperature sensor, converted by group injected only. This internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.

LL_ADC_AWD_CH_TEMPSENSOR_REG_INJ

ADC analog watchdog monitoring of ADC internal channel connected to Temperature sensor, converted by either group regular or injected. This internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.

Analog watchdog - Analog watchdog number

LL_ADC_AWD1

ADC analog watchdog number 1

Analog watchdog - Thresholds

LL_ADC_AWD_THRESHOLD_HIGH

ADC analog watchdog threshold high

LL_ADC_AWD_THRESHOLD_LOW

ADC analog watchdog threshold low

ADC instance - Channel number

LL_ADC_CHANNEL_0

ADC external channel (channel connected to GPIO pin) ADCx_IN0

LL_ADC_CHANNEL_1

ADC external channel (channel connected to GPIO pin) ADCx_IN1

LL_ADC_CHANNEL_2

ADC external channel (channel connected to GPIO pin) ADCx_IN2

LL_ADC_CHANNEL_3

ADC external channel (channel connected to GPIO pin) ADCx_IN3

LL_ADC_CHANNEL_4

ADC external channel (channel connected to GPIO pin) ADCx_IN4

LL_ADC_CHANNEL_5

ADC external channel (channel connected to GPIO pin) ADCx_IN5

LL_ADC_CHANNEL_6

ADC external channel (channel connected to GPIO pin) ADCx_IN6

LL_ADC_CHANNEL_7

ADC external channel (channel connected to GPIO pin) ADCx_IN7

LL_ADC_CHANNEL_8

ADC external channel (channel connected to GPIO pin) ADCx_IN8

LL_ADC_CHANNEL_9

ADC external channel (channel connected to GPIO pin) ADCx_IN9

LL_ADC_CHANNEL_10

ADC external channel (channel connected to GPIO pin) ADCx_IN10

LL_ADC_CHANNEL_11

ADC external channel (channel connected to GPIO pin) ADCx_IN11

LL_ADC_CHANNEL_12

ADC external channel (channel connected to GPIO pin) ADCx_IN12

LL_ADC_CHANNEL_13

ADC external channel (channel connected to GPIO pin) ADCx_IN13

LL_ADC_CHANNEL_14

ADC external channel (channel connected to GPIO pin) ADCx_IN14

LL_ADC_CHANNEL_15

ADC external channel (channel connected to GPIO pin) ADCx_IN15

LL_ADC_CHANNEL_16

ADC external channel (channel connected to GPIO pin) ADCx_IN16

LL_ADC_CHANNEL_17

ADC external channel (channel connected to GPIO pin) ADCx_IN17

LL_ADC_CHANNEL_18

ADC external channel (channel connected to GPIO pin) ADCx_IN18

LL_ADC_CHANNEL_VREFINT

ADC internal channel connected to VrefInt: Internal voltage reference. On STM32F4, ADC channel available only on ADC instance: ADC1.

LL_ADC_CHANNEL_VBAT

ADC internal channel connected to Vbat/3: Vbat voltage through a divider ladder of factor 1/3 to have Vbat always below Vdda. On STM32F4, ADC channel available only on ADC instance: ADC1.

LL_ADC_CHANNEL_TEMPSENSOR

ADC internal channel connected to Temperature sensor. On STM32F4, ADC channel available only on ADC instance: ADC1. This internal channel is shared between temperature sensor and Vbat, only 1 measurement path must be enabled.

Channel - Sampling time

LL_ADC_SAMPLINGTIME_3CYCLES

Sampling time 3 ADC clock cycles

LL_ADC_SAMPLINGTIME_15CYCLES

Sampling time 15 ADC clock cycles

LL_ADC_SAMPLINGTIME_28CYCLES

Sampling time 28 ADC clock cycles

LL_ADC_SAMPLINGTIME_56CYCLES

Sampling time 56 ADC clock cycles

LL_ADC_SAMPLINGTIME_84CYCLES

Sampling time 84 ADC clock cycles

LL_ADC_SAMPLINGTIME_112CYCLES

Sampling time 112 ADC clock cycles

LL_ADC_SAMPLINGTIME_144CYCLES

Sampling time 144 ADC clock cycles

LL_ADC_SAMPLINGTIME_480CYCLES

Sampling time 480 ADC clock cycles

ADC common - Clock source

LL_ADC_CLOCK_SYNC_PCLK_DIV2

ADC synchronous clock derived from AHB clock with prescaler division by 2

LL_ADC_CLOCK_SYNC_PCLK_DIV4

ADC synchronous clock derived from AHB clock with prescaler division by 4

LL_ADC_CLOCK_SYNC_PCLK_DIV6

ADC synchronous clock derived from AHB clock with prescaler division by 6

LL_ADC_CLOCK_SYNC_PCLK_DIV8

ADC synchronous clock derived from AHB clock with prescaler division by 8

ADC common - Measurement path to internal channels

LL_ADC_PATH_INTERNAL_NONE

ADC measurement pathes all disabled

LL_ADC_PATH_INTERNAL_VREFINT

ADC measurement path to internal channel VrefInt

LL_ADC_PATH_INTERNAL_TEMPSENSOR

ADC measurement path to internal channel temperature sensor

LL_ADC_PATH_INTERNAL_VBAT

ADC measurement path to internal channel Vbat

ADC instance - Data alignment

LL_ADC_DATA_ALIGN_RIGHT

ADC conversion data alignment: right aligned (alignment on data register LSB bit 0)

LL_ADC_DATA_ALIGN_LEFT

ADC conversion data alignment: left aligned (aligment on data register MSB bit 15)

ADC flags

LL_ADC_FLAG_STRT

ADC flag ADC group regular conversion start

LL_ADC_FLAG_EOCS

ADC flag ADC group regular end of unitary conversion or sequence conversions (to configure flag of end of conversion, use function

LL_ADC_FLAG_OVR

ADC flag ADC group regular overrun

LL_ADC_FLAG_JSTRT

ADC flag ADC group injected conversion start

LL_ADC_FLAG_JEOS

ADC flag ADC group injected end of sequence conversions (Note: on this STM32 serie, there is no flag ADC group injected end of unitary conversion. Flag noted as "JEOC" is corresponding to flag "JEOS" in other STM32 families)

LL_ADC_FLAG_AWD1

ADC flag ADC analog watchdog 1

LL_ADC_FLAG_EOCS_MST

ADC flag ADC multimode master group regular end of unitary conversion or sequence conversions (to configure flag of end of conversion, use function

LL_ADC_FLAG_EOCS_SLV1

ADC flag ADC multimode slave 1 group regular end of unitary conversion or sequence conversions (to configure flag of end of conversion, use function

LL_ADC_FLAG_EOCS_SLV2

ADC flag ADC multimode slave 2 group regular end of unitary conversion or sequence conversions (to configure flag of end of conversion, use function

LL_ADC_FLAG_OVR_MST

ADC flag ADC multimode master group regular overrun

LL_ADC_FLAG_OVR_SLV1

ADC flag ADC multimode slave 1 group regular overrun

LL_ADC_FLAG_OVR_SLV2

ADC flag ADC multimode slave 2 group regular overrun

LL_ADC_FLAG_JEOS_MST

ADC flag ADC multimode master group injected end of sequence conversions (Note: on this STM32 serie, there is no flag ADC group injected end of unitary conversion. Flag noted as "JEOC" is corresponding to flag "JEOS" in other STM32 families)

LL_ADC_FLAG_JEOS_SLV1

ADC flag ADC multimode slave 1 group injected end of sequence conversions (Note: on this STM32 serie, there is no flag ADC group injected end of unitary conversion. Flag noted as "JEOC" is corresponding to flag "JEOS" in other STM32 families)

LL_ADC_FLAG_JEOS_SLV2

ADC flag ADC multimode slave 2 group injected end of sequence conversions (Note: on this STM32 serie, there is no flag ADC group injected end of unitary conversion. Flag noted as "JEOC" is corresponding to flag "JEOS" in other STM32 families)

LL_ADC_FLAG_AWD1_MST

ADC flag ADC multimode master analog watchdog 1 of the ADC master

LL_ADC_FLAG_AWD1_SLV1

ADC flag ADC multimode slave 1 analog watchdog 1

LL_ADC_FLAG_AWD1_SLV2

ADC flag ADC multimode slave 2 analog watchdog 1

ADC instance - Groups

LL_ADC_GROUP_REGULAR

ADC group regular (available on all STM32 devices)

LL_ADC_GROUP_INJECTED

ADC group injected (not available on all STM32 devices)

LL_ADC_GROUP_REGULAR_INJECTED

ADC both groups regular and injected

Definitions of ADC hardware constraints delays

LL_ADC_DELAY_VREFINT_STAB_US

Delay for internal voltage reference stabilization time

LL_ADC_DELAY_TEMPSENSOR_STAB_US

Delay for internal voltage reference stabilization time

ADC group injected - Sequencer discontinuous mode

LL_ADC_INJ_SEQ_DISCONT_DISABLE

ADC group injected sequencer discontinuous mode disable

LL_ADC_INJ_SEQ_DISCONT_1RANK

ADC group injected sequencer discontinuous mode enable with sequence interruption every rank

ADC group injected - Sequencer ranks

LL_ADC_INJ_RANK_1

ADC group injected sequencer rank 1

LL_ADC_INJ_RANK_2

ADC group injected sequencer rank 2

LL_ADC_INJ_RANK_3

ADC group injected sequencer rank 3

LL_ADC_INJ_RANK_4

ADC group injected sequencer rank 4

ADC group injected - Sequencer scan length

LL_ADC_INJ_SEQ_SCAN_DISABLE

ADC group injected sequencer disable (equivalent to sequencer of 1 rank: ADC conversion on only 1 channel)

LL_ADC_INJ_SEQ_SCAN_ENABLE_2RANKS

ADC group injected sequencer enable with 2 ranks in the sequence

LL_ADC_INJ_SEQ_SCAN_ENABLE_3RANKS

ADC group injected sequencer enable with 3 ranks in the sequence

LL_ADC_INJ_SEQ_SCAN_ENABLE_4RANKS

ADC group injected sequencer enable with 4 ranks in the sequence

ADC group injected - Trigger edge

LL_ADC_INJ_TRIG_EXT_RISING

ADC group injected conversion trigger polarity set to rising edge

LL_ADC_INJ_TRIG_EXT_FALLING

ADC group injected conversion trigger polarity set to falling edge

LL_ADC_INJ_TRIG_EXT_RISINGFALLING

ADC group injected conversion trigger polarity set to both rising and falling edges

ADC group injected - Trigger source

LL_ADC_INJ_TRIG_SOFTWARE

ADC group injected conversion trigger internal: SW start.

LL_ADC_INJ_TRIG_EXT_TIM1_CH4

ADC group injected conversion trigger from external IP: TIM1 channel 4 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM1_TRGO

ADC group injected conversion trigger from external IP: TIM1 TRGO. Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM2_CH1

ADC group injected conversion trigger from external IP: TIM2 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM2_TRGO

ADC group injected conversion trigger from external IP: TIM2 TRGO. Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM3_CH2

ADC group injected conversion trigger from external IP: TIM3 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM3_CH4

ADC group injected conversion trigger from external IP: TIM3 channel 4 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM4_CH1

ADC group injected conversion trigger from external IP: TIM4 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM4_CH2

ADC group injected conversion trigger from external IP: TIM4 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM4_CH3

ADC group injected conversion trigger from external IP: TIM4 channel 3 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM4_TRGO

ADC group injected conversion trigger from external IP: TIM4 TRGO. Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM5_CH4

ADC group injected conversion trigger from external IP: TIM5 channel 4 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM5_TRGO

ADC group injected conversion trigger from external IP: TIM5 TRGO. Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM8_CH2

ADC group injected conversion trigger from external IP: TIM8 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM8_CH3

ADC group injected conversion trigger from external IP: TIM8 channel 3 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_TIM8_CH4

ADC group injected conversion trigger from external IP: TIM8 channel 4 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_INJ_TRIG_EXT_EXTI_LINE15

ADC group injected conversion trigger from external IP: external interrupt line 15. Trigger edge set to rising edge (default setting).

ADC group injected - Automatic trigger mode

LL_ADC_INJ_TRIG_INDEPENDENT

ADC group injected conversion trigger independent. Setting mandatory if ADC group injected injected trigger source is set to an external trigger.

LL_ADC_INJ_TRIG_FROM_GRP_REGULAR

ADC group injected conversion trigger from ADC group regular. Setting compliant only with group injected trigger source set to SW start, without any further action on ADC group injected conversion start or stop: in this case, ADC group injected is controlled only from ADC group regular.

ADC interruptions for configuration (interruption enable or disable)

LL_ADC_IT_EOCS

ADC interruption ADC group regular end of unitary conversion or sequence conversions (to configure flag of end of conversion, use function

LL_ADC_IT_OVR

ADC interruption ADC group regular overrun

LL_ADC_IT_JEOS

ADC interruption ADC group injected end of sequence conversions (Note: on this STM32 serie, there is no flag ADC group injected end of unitary conversion. Flag noted as "JEOC" is corresponding to flag "JEOS" in other STM32 families)

LL_ADC_IT_AWD1

ADC interruption ADC analog watchdog 1

Multimode - DMA transfer

LL_ADC_MULTI_REG_DMA_EACH_ADC

ADC multimode group regular conversions are transferred by DMA: each ADC uses its own DMA channel, with its individual DMA transfer settings

LL_ADC_MULTI_REG_DMA_LIMIT_1

ADC multimode group regular conversions are transferred by DMA, one DMA channel for all ADC instances (DMA of ADC master), in limited mode (one shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular. Setting of DMA mode 1: 2 or 3 (dual or triple mode) half-words one by one, ADC1 then ADC2 then ADC3.

LL_ADC_MULTI_REG_DMA_LIMIT_2

ADC multimode group regular conversions are transferred by DMA, one DMA channel for all ADC instances (DMA of ADC master), in limited mode (one shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular. Setting of DMA mode 2: 2 or 3 (dual or triple mode) half-words one by one, ADC2&1 then ADC1&3 then ADC3&2.

LL_ADC_MULTI_REG_DMA_LIMIT_3

ADC multimode group regular conversions are transferred by DMA, one DMA channel for all ADC instances (DMA of ADC master), in limited mode (one shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular. Setting of DMA mode 3: 2 or 3 (dual or triple mode) bytes one by one, ADC2&1 then ADC1&3 then ADC3&2.

LL_ADC_MULTI_REG_DMA_UNLMT_1

ADC multimode group regular conversions are transferred by DMA, one DMA channel for all ADC instances (DMA of ADC master), in unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transferred (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular. Setting of DMA mode 1: 2 or 3 (dual or triple mode) half-words one by one, ADC1 then ADC2 then ADC3.

LL_ADC_MULTI_REG_DMA_UNLMT_2

ADC multimode group regular conversions are transferred by DMA, one DMA channel for all ADC instances (DMA of ADC master), in unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transferred (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular. Setting of DMA mode 2: 2 or 3 (dual or triple mode) half-words by pairs, ADC2&1 then ADC1&3 then ADC3&2.

LL_ADC_MULTI_REG_DMA_UNLMT_3

ADC multimode group regular conversions are transferred by DMA, one DMA channel for all ADC instances (DMA of ADC master), in unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transferred (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular. Setting of DMA mode 3: 2 or 3 (dual or triple mode) bytes one by one, ADC2&1 then ADC1&3 then ADC3&2.

Multimode - ADC master or slave

LL_ADC_MULTI_MASTER

In multimode, selection among several ADC instances: ADC master

LL_ADC_MULTI_SLAVE

In multimode, selection among several ADC instances: ADC slave

LL_ADC_MULTI_MASTER_SLAVE

In multimode, selection among several ADC instances: both ADC master and ADC slave

Multimode - Mode

LL_ADC_MULTI_INDEPENDENT

ADC dual mode disabled (ADC independent mode)

LL_ADC_MULTI_DUAL_REG_SIMULT

ADC dual mode enabled: group regular simultaneous

LL_ADC_MULTI_DUAL_REG_INTERL

ADC dual mode enabled: Combined group regular interleaved

LL_ADC_MULTI_DUAL_INJ_SIMULT

ADC dual mode enabled: group injected simultaneous

LL_ADC_MULTI_DUAL_INJ_ALTERN

ADC dual mode enabled: group injected alternate trigger. Works only with external triggers (not internal SW start)

LL_ADC_MULTI_DUAL_REG_SIM_INJ_SIM

ADC dual mode enabled: Combined group regular simultaneous + group injected simultaneous

LL_ADC_MULTI_DUAL_REG_SIM_INJ_ALT

ADC dual mode enabled: Combined group regular simultaneous + group injected alternate trigger

LL_ADC_MULTI_DUAL_REG_INT_INJ_SIM

ADC dual mode enabled: Combined group regular interleaved + group injected simultaneous

LL_ADC_MULTI_TRIPLE_REG_SIM_INJ_SIM

ADC triple mode enabled: Combined group regular simultaneous + group injected simultaneous

LL_ADC_MULTI_TRIPLE_REG_SIM_INJ_ALT

ADC triple mode enabled: Combined group regular simultaneous + group injected alternate trigger

LL_ADC_MULTI_TRIPLE_INJ_SIMULT

ADC triple mode enabled: group injected simultaneous

LL_ADC_MULTI_TRIPLE_REG_SIMULT

ADC triple mode enabled: group regular simultaneous

LL_ADC_MULTI_TRIPLE_REG_INTERL

ADC triple mode enabled: Combined group regular interleaved

LL_ADC_MULTI_TRIPLE_INJ_ALTERN

ADC triple mode enabled: group injected alternate trigger. Works only with external triggers (not internal SW start)

Multimode - Delay between two sampling phases

LL_ADC_MULTI_TWOSMP_DELAY_5CYCLES

ADC multimode delay between two sampling phases: 5 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_6CYCLES

ADC multimode delay between two sampling phases: 6 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_7CYCLES

ADC multimode delay between two sampling phases: 7 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_8CYCLES

ADC multimode delay between two sampling phases: 8 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_9CYCLES

ADC multimode delay between two sampling phases: 9 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_10CYCLES

ADC multimode delay between two sampling phases: 10 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_11CYCLES

ADC multimode delay between two sampling phases: 11 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_12CYCLES

ADC multimode delay between two sampling phases: 12 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_13CYCLES

ADC multimode delay between two sampling phases: 13 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_14CYCLES

ADC multimode delay between two sampling phases: 14 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_15CYCLES

ADC multimode delay between two sampling phases: 15 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_16CYCLES

ADC multimode delay between two sampling phases: 16 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_17CYCLES

ADC multimode delay between two sampling phases: 17 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_18CYCLES

ADC multimode delay between two sampling phases: 18 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_19CYCLES

ADC multimode delay between two sampling phases: 19 ADC clock cycles

LL_ADC_MULTI_TWOSMP_DELAY_20CYCLES

ADC multimode delay between two sampling phases: 20 ADC clock cycles

ADC registers compliant with specific purpose

LL_ADC_DMA_REG_REGULAR_DATA

LL_ADC_DMA_REG_REGULAR_DATA_MULTI

ADC group regular - Continuous mode

LL_ADC_REG_CONV_SINGLE

ADC conversions are performed in single mode: one conversion per trigger

LL_ADC_REG_CONV_CONTINUOUS

ADC conversions are performed in continuous mode: after the first trigger, following conversions launched successively automatically

ADC group regular - DMA transfer of ADC conversion data

LL_ADC_REG_DMA_TRANSFER_NONE

ADC conversions are not transferred by DMA

LL_ADC_REG_DMA_TRANSFER_LIMITED

ADC conversion data are transferred by DMA, in limited mode (one shot mode): DMA transfer requests are stopped when number of DMA data transfers (number of ADC conversions) is reached. This ADC mode is intended to be used with DMA mode non-circular.

LL_ADC_REG_DMA_TRANSFER_UNLIMITED

ADC conversion data are transferred by DMA, in unlimited mode: DMA transfer requests are unlimited, whatever number of DMA data transferred (number of ADC conversions). This ADC mode is intended to be used with DMA mode circular.

ADC group regular - Flag EOC selection (unitary or sequence conversions)

LL_ADC_REG_FLAG_EOC_SEQUENCE_CONV

ADC flag EOC (end of unitary conversion) selected

LL_ADC_REG_FLAG_EOC_UNITARY_CONV

ADC flag EOS (end of sequence conversions) selected

ADC group regular - Sequencer discontinuous mode

LL_ADC_REG_SEQ_DISCONT_DISABLE

ADC group regular sequencer discontinuous mode disable

LL_ADC_REG_SEQ_DISCONT_1RANK

ADC group regular sequencer discontinuous mode enable with sequence interruption every rank

LL_ADC_REG_SEQ_DISCONT_2RANKS

ADC group regular sequencer discontinuous mode enabled with sequence interruption every 2 ranks

LL_ADC_REG_SEQ_DISCONT_3RANKS

ADC group regular sequencer discontinuous mode enable with sequence interruption every 3 ranks

LL_ADC_REG_SEQ_DISCONT_4RANKS

ADC group regular sequencer discontinuous mode enable with sequence interruption every 4 ranks

LL_ADC_REG_SEQ_DISCONT_5RANKS

ADC group regular sequencer discontinuous mode enable with sequence interruption every 5 ranks

LL_ADC_REG_SEQ_DISCONT_6RANKS

ADC group regular sequencer discontinuous mode enable with sequence interruption every 6 ranks

LL_ADC_REG_SEQ_DISCONT_7RANKS

ADC group regular sequencer discontinuous mode enable with sequence interruption every 7 ranks

LL_ADC_REG_SEQ_DISCONT_8RANKS

ADC group regular sequencer discontinuous mode enable with sequence interruption every 8 ranks

ADC group regular - Sequencer ranks

LL_ADC_REG_RANK_1

ADC group regular sequencer rank 1

LL_ADC_REG_RANK_2

ADC group regular sequencer rank 2

LL_ADC_REG_RANK_3

ADC group regular sequencer rank 3

LL_ADC_REG_RANK_4

ADC group regular sequencer rank 4

LL_ADC_REG_RANK_5

ADC group regular sequencer rank 5

LL_ADC_REG_RANK_6

ADC group regular sequencer rank 6

LL_ADC_REG_RANK_7

ADC group regular sequencer rank 7

LL_ADC_REG_RANK_8

ADC group regular sequencer rank 8

LL_ADC_REG_RANK_9

ADC group regular sequencer rank 9

LL_ADC_REG_RANK_10

ADC group regular sequencer rank 10

LL_ADC_REG_RANK_11

ADC group regular sequencer rank 11

LL_ADC_REG_RANK_12

ADC group regular sequencer rank 12

LL_ADC_REG_RANK_13

ADC group regular sequencer rank 13

LL_ADC_REG_RANK_14

ADC group regular sequencer rank 14

LL_ADC_REG_RANK_15

ADC group regular sequencer rank 15

LL_ADC_REG_RANK_16

ADC group regular sequencer rank 16

ADC group regular - Sequencer scan length

LL_ADC_REG_SEQ_SCAN_DISABLE

ADC group regular sequencer disable (equivalent to sequencer of 1 rank: ADC conversion on only 1 channel)

LL_ADC_REG_SEQ_SCAN_ENABLE_2RANKS

ADC group regular sequencer enable with 2 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_3RANKS

ADC group regular sequencer enable with 3 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_4RANKS

ADC group regular sequencer enable with 4 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_5RANKS

ADC group regular sequencer enable with 5 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_6RANKS

ADC group regular sequencer enable with 6 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_7RANKS

ADC group regular sequencer enable with 7 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_8RANKS

ADC group regular sequencer enable with 8 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_9RANKS

ADC group regular sequencer enable with 9 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_10RANKS

ADC group regular sequencer enable with 10 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_11RANKS

ADC group regular sequencer enable with 11 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_12RANKS

ADC group regular sequencer enable with 12 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_13RANKS

ADC group regular sequencer enable with 13 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_14RANKS

ADC group regular sequencer enable with 14 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_15RANKS

ADC group regular sequencer enable with 15 ranks in the sequence

LL_ADC_REG_SEQ_SCAN_ENABLE_16RANKS

ADC group regular sequencer enable with 16 ranks in the sequence

ADC group regular - Trigger edge

LL_ADC_REG_TRIG_EXT_RISING

ADC group regular conversion trigger polarity set to rising edge

LL_ADC_REG_TRIG_EXT_FALLING

ADC group regular conversion trigger polarity set to falling edge

LL_ADC_REG_TRIG_EXT_RISINGFALLING

ADC group regular conversion trigger polarity set to both rising and falling edges

ADC group regular - Trigger source

LL_ADC_REG_TRIG_SOFTWARE

ADC group regular conversion trigger internal: SW start.

LL_ADC_REG_TRIG_EXT_TIM1_CH1

ADC group regular conversion trigger from external IP: TIM1 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM1_CH2

ADC group regular conversion trigger from external IP: TIM1 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM1_CH3

ADC group regular conversion trigger from external IP: TIM1 channel 3 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM2_CH2

ADC group regular conversion trigger from external IP: TIM2 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM2_CH3

ADC group regular conversion trigger from external IP: TIM2 channel 3 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM2_CH4

ADC group regular conversion trigger from external IP: TIM2 channel 4 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM2_TRGO

ADC group regular conversion trigger from external IP: TIM2 TRGO. Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM3_CH1

ADC group regular conversion trigger from external IP: TIM3 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM3_TRGO

ADC group regular conversion trigger from external IP: TIM3 TRGO. Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM4_CH4

ADC group regular conversion trigger from external IP: TIM4 channel 4 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM5_CH1

ADC group regular conversion trigger from external IP: TIM5 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM5_CH2

ADC group regular conversion trigger from external IP: TIM5 channel 2 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM5_CH3

ADC group regular conversion trigger from external IP: TIM5 channel 3 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM8_CH1

ADC group regular conversion trigger from external IP: TIM8 channel 1 event (capture compare: input capture or output capture). Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_TIM8_TRGO

ADC group regular conversion trigger from external IP: TIM8 TRGO. Trigger edge set to rising edge (default setting).

LL_ADC_REG_TRIG_EXT_EXTI_LINE11

ADC group regular conversion trigger from external IP: external interrupt line 11. Trigger edge set to rising edge (default setting).

ADC instance - Resolution

LL_ADC_RESOLUTION_12B

ADC resolution 12 bits

LL_ADC_RESOLUTION_10B

ADC resolution 10 bits

LL_ADC_RESOLUTION_8B

ADC resolution 8 bits

LL_ADC_RESOLUTION_6B

ADC resolution 6 bits

ADC instance - Scan selection

LL_ADC_SEQ_SCAN_DISABLE

ADC conversion is performed in unitary conversion mode (one channel converted, that defined in rank 1). Configuration of both groups regular and injected sequencers (sequence length, ...) is discarded: equivalent to length of 1 rank.

LL_ADC_SEQ_SCAN_ENABLE

ADC conversions are performed in sequence conversions mode, according to configuration of both groups regular and injected sequencers (sequence length, ...).

ADC helper macro

__LL_ADC_CHANNEL_TO_DECIMAL_NB

Description:

- Helper macro to get ADC channel number in decimal format from literals LL_ADC_CHANNEL_x.

Parameters:

- __CHANNEL__: This parameter can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

Return value:

- Value: between Min_Data=0 and Max_Data=18

Notes:

- Example: __LL_ADC_CHANNEL_TO_DECIMAL_NB(LL_ADC_CHANNEL_4) will return decimal number "4". The input can be a value from functions where a channel number is returned, either defined with number or with bitfield (only one bit must be set).

__LL_ADC_DECIMAL_NB_TO_CHANNEL

Description:

- Helper macro to get ADC channel in literal format LL_ADC_CHANNEL_x from number in decimal format.

Parameters:

- __DECIMAL_NB__: Value between Min_Data=0 and Max_Data=18

Return value:

- Returned: value can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

Notes:

- Example: __LL_ADC_DECIMAL_NB_TO_CHANNEL(4) will return a data equivalent to "LL_ADC_CHANNEL_4".

__LL_ADC_IS_CHANNEL_INTERNAL

Description:

- Helper macro to determine whether the selected channel corresponds to literal definitions of driver.

Parameters:

- __CHANNEL__: This parameter can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

Return value:

- Value: "0" if the channel corresponds to a parameter definition of a ADC external channel (channel connected to a GPIO pin). Value "1" if the channel corresponds to a parameter definition of a ADC internal channel.

Notes:

- The different literal definitions of ADC channels are: ADC internal channel: LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, ...ADC external channel (channel connected to a GPIO pin): LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, ... The channel parameter must be a value defined from literal definition of a ADC internal channel (LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, ...), ADC external channel (LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, ...), must not be a value from functions where a channel number is returned from ADC registers, because internal and external channels share the same channel number in ADC registers. The differentiation is made only with parameters definitions of driver.

__LL_ADC_CHANNEL_INTERNAL_TO_EXTERNAL

Description:

- Helper macro to convert a channel defined from parameter definition of a ADC internal channel (LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, ...), to its equivalent parameter definition of a ADC external channel (LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, ...).

Parameters:

- __CHANNEL__: This parameter can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

Return value:

- Returned: value can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18

Notes:

- The channel parameter can be, additionally to a value defined from parameter definition of a ADC internal channel (LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, ...), a value defined from parameter definition of ADC external channel (LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, ...) or a value from functions where a channel number is returned from ADC registers.

__LL_ADC_IS_CHANNEL_INTERNAL_AVAILABLE

Description:

- Helper macro to determine whether the internal channel selected is available on the ADC instance selected.

Parameters:

- __ADC_INSTANCE__: ADC instance
- __CHANNEL__: This parameter can be one of the following values:
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)

Return value:

- Value: "0" if the internal channel selected is not available on the ADC instance selected. Value "1" if the internal channel selected is available on the ADC instance selected.

Notes:

- The channel parameter must be a value defined from parameter definition of a ADC internal channel (LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, ...), must not be a value defined from parameter definition of ADC external channel (LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, ...) or a value from functions where a channel number is returned from ADC registers, because internal and external channels share the same channel number in ADC registers. The differentiation is made only with parameters definitions of driver.

__LL_ADC_ANALOGWD_CHANNEL_GROUP

Description:

- Helper macro to define ADC analog watchdog parameter: define a single channel to monitor with analog watchdog from sequencer channel and groups definition.

Parameters:

- __CHANNEL__: This parameter can be one of the following values:
  - LL_ADC_CHANNEL_0
  - LL_ADC_CHANNEL_1
  - LL_ADC_CHANNEL_2
  - LL_ADC_CHANNEL_3
  - LL_ADC_CHANNEL_4
  - LL_ADC_CHANNEL_5
  - LL_ADC_CHANNEL_6
  - LL_ADC_CHANNEL_7
  - LL_ADC_CHANNEL_8
  - LL_ADC_CHANNEL_9
  - LL_ADC_CHANNEL_10
  - LL_ADC_CHANNEL_11
  - LL_ADC_CHANNEL_12
  - LL_ADC_CHANNEL_13
  - LL_ADC_CHANNEL_14
  - LL_ADC_CHANNEL_15
  - LL_ADC_CHANNEL_16
  - LL_ADC_CHANNEL_17
  - LL_ADC_CHANNEL_18
  - LL_ADC_CHANNEL_VREFINT (1)
  - LL_ADC_CHANNEL_TEMPSENSOR (1)(2)
  - LL_ADC_CHANNEL_VBAT (1)
- __GROUP__: This parameter can be one of the following values:
  - LL_ADC_GROUP_REGULAR
  - LL_ADC_GROUP_INJECTED
  - LL_ADC_GROUP_REGULAR_INJECTED

Return value:

- Returned: value can be one of the following values:
  - LL_ADC_AWD_DISABLE
  - LL_ADC_AWD_ALL_CHANNELS_REG
  - LL_ADC_AWD_ALL_CHANNELS_INJ
  - LL_ADC_AWD_ALL_CHANNELS_REG_INJ
  - LL_ADC_AWD_CHANNEL_0_REG
  - LL_ADC_AWD_CHANNEL_0_INJ
  - LL_ADC_AWD_CHANNEL_0_REG_INJ
  - LL_ADC_AWD_CHANNEL_1_REG
  - LL_ADC_AWD_CHANNEL_1_INJ
  - LL_ADC_AWD_CHANNEL_1_REG_INJ
  - LL_ADC_AWD_CHANNEL_2_REG
  - LL_ADC_AWD_CHANNEL_2_INJ
  - LL_ADC_AWD_CHANNEL_2_REG_INJ
  - LL_ADC_AWD_CHANNEL_3_REG
  - LL_ADC_AWD_CHANNEL_3_INJ
  - LL_ADC_AWD_CHANNEL_3_REG_INJ
  - LL_ADC_AWD_CHANNEL_4_REG
  - LL_ADC_AWD_CHANNEL_4_INJ
  - LL_ADC_AWD_CHANNEL_4_REG_INJ
  - LL_ADC_AWD_CHANNEL_5_REG
  - LL_ADC_AWD_CHANNEL_5_INJ
  - LL_ADC_AWD_CHANNEL_5_REG_INJ
  - LL_ADC_AWD_CHANNEL_6_REG
  - LL_ADC_AWD_CHANNEL_6_INJ
  - LL_ADC_AWD_CHANNEL_6_REG_INJ
  - LL_ADC_AWD_CHANNEL_7_REG
  - LL_ADC_AWD_CHANNEL_7_INJ
  - LL_ADC_AWD_CHANNEL_7_REG_INJ
  - LL_ADC_AWD_CHANNEL_8_REG
  - LL_ADC_AWD_CHANNEL_8_INJ
  - LL_ADC_AWD_CHANNEL_8_REG_INJ
  - LL_ADC_AWD_CHANNEL_9_REG
  - LL_ADC_AWD_CHANNEL_9_INJ
  - LL_ADC_AWD_CHANNEL_9_REG_INJ
  - LL_ADC_AWD_CHANNEL_10_REG
  - LL_ADC_AWD_CHANNEL_10_INJ
  - LL_ADC_AWD_CHANNEL_10_REG_INJ
  - LL_ADC_AWD_CHANNEL_11_REG
  - LL_ADC_AWD_CHANNEL_11_INJ
  - LL_ADC_AWD_CHANNEL_11_REG_INJ
  - LL_ADC_AWD_CHANNEL_12_REG
  - LL_ADC_AWD_CHANNEL_12_INJ
  - LL_ADC_AWD_CHANNEL_12_REG_INJ
  - LL_ADC_AWD_CHANNEL_13_REG
  - LL_ADC_AWD_CHANNEL_13_INJ
  - LL_ADC_AWD_CHANNEL_13_REG_INJ
  - LL_ADC_AWD_CHANNEL_14_REG
  - LL_ADC_AWD_CHANNEL_14_INJ
  - LL_ADC_AWD_CHANNEL_14_REG_INJ
  - LL_ADC_AWD_CHANNEL_15_REG
  - LL_ADC_AWD_CHANNEL_15_INJ

Notes:

- To be used with function LL_ADC_SetAnalogWDMonitChannels().
  Example: LL_ADC_SetAnalogWDMonitChannels(ADC1, LL_ADC_AWD1,
  __LL_ADC_ANALOGWD_CHANNEL_GROUP(LL_ADC_CHANNEL4, LL_ADC_GROUP_REGULAR))

__LL_ADC_ANALOGWD_SET_THRESHOLD_RESOLUTION

Description:

- Helper macro to set the value of ADC analog watchdog threshold high or low in function of ADC resolution, when ADC resolution is different of 12 bits.

Parameters:

- __ADC_RESOLUTION__: This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B
- __AWD_THRESHOLD__: Value between Min_Data=0x000 and Max_Data=0xFFF

Return value:

- Value: between Min_Data=0x000 and Max_Data=0xFFF

Notes:

- To be used with function LL_ADC_SetAnalogWDThresholds(). Example, with a ADC resolution of 8 bits, to set the value of analog watchdog threshold high (on 8 bits): LL_ADC_SetAnalogWDThresholds (< ADCx param >, __LL_ADC_ANALOGWD_SET_THRESHOLD_RESOLUTION(LL_ADC_RESOLUTION_8B, <threshold_value_8_bits>));

__LL_ADC_ANALOGWD_GET_THRESHOLD_RESOLUTION

Description:

- Helper macro to get the value of ADC analog watchdog threshold high or low in function of ADC resolution, when ADC resolution is different of 12 bits.

Parameters:

- __ADC_RESOLUTION__: This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B
- __AWD_THRESHOLD_12_BITS__: Value between Min_Data=0x000 and Max_Data=0xFFF

Return value:

- Value: between Min_Data=0x000 and Max_Data=0xFFF

Notes:

- To be used with function LL_ADC_GetAnalogWDThresholds(). Example, with a ADC resolution of 8 bits, to get the value of analog watchdog threshold high (on 8 bits): < threshold_value_6_bits > = __LL_ADC_ANALOGWD_GET_THRESHOLD_RESOLUTION (LL_ADC_RESOLUTION_8B, LL_ADC_GetAnalogWDThresholds(<ADCx param>, LL_ADC_AWD_THRESHOLD_HIGH));

__LL_ADC_MULTI_CONV_DATA_MASTER_SLAVE

Description:

- Helper macro to get the ADC multimode conversion data of ADC master or ADC slave from raw value with both ADC conversion data concatenated.

Parameters:

- __ADC_MULTI_MASTER_SLAVE__: This parameter can be one of the following values:
  - LL_ADC_MULTI_MASTER
  - LL_ADC_MULTI_SLAVE
- __ADC_MULTI_CONV_DATA__: Value between Min_Data=0x000 and Max_Data=0xFFF

Return value:

- Value: between Min_Data=0x000 and Max_Data=0xFFF

Notes:

- This macro is intended to be used when multimode transfer by DMA is enabled: refer to function LL_ADC_SetMultiDMATransfer(). In this case the transferred data need to processed with this macro to separate the conversion data of ADC master and ADC slave.

__LL_ADC_COMMON_INSTANCE

Description:

- Helper macro to select the ADC common instance to which is belonging the selected ADC instance.

Parameters:

- __ADCx__: ADC instance

Return value:

- ADC: common register instance

Notes:

- ADC common register instance can be used for: Set parameters common to several ADC instancesMultimode (for devices with several ADC instances) Refer to functions having argument "ADCxy_COMMON" as parameter.

__LL_ADC_IS_ENABLED_ALL_COMMON_INSTANCE

Description:

- Helper macro to check if all ADC instances sharing the same ADC common instance are disabled.

Parameters:

- __ADCXY_COMMON__: ADC common instance (can be set directly from CMSIS definition or by using helper macro

Return value:

- Value: "0" if all ADC instances sharing the same ADC common instance are disabled. Value "1" if at least one ADC instance sharing the same ADC common instance is enabled.

Notes:

- This check is required by functions with setting conditioned to ADC state: All ADC instances of the ADC common group must be disabled. Refer to functions having argument "ADCxy_COMMON" as parameter. On devices with only 1 ADC common instance, parameter of this macro is useless and can be ignored (parameter kept for compatibility with devices featuring several ADC common instances).

__LL_ADC_DIGITAL_SCALE

Description:

- Helper macro to define the ADC conversion data full-scale digital value corresponding to the selected ADC resolution.

Parameters:

- __ADC_RESOLUTION__: This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B

Return value:

- ADC: conversion data equivalent voltage value (unit: mVolt)

Notes:

- ADC conversion data full-scale corresponds to voltage range determined by analog voltage references Vref+ and Vref- (refer to reference manual).

__LL_ADC_CONVERT_DATA_RESOLUTION

Description:

- Helper macro to convert the ADC conversion data from a resolution to another resolution.

Parameters:

- __DATA__: ADC conversion data to be converted
- __ADC_RESOLUTION_CURRENT__: Resolution of to the data to be converted This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B
- __ADC_RESOLUTION_TARGET__: Resolution of the data after conversion This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B

Return value:

- ADC: conversion data to the requested resolution

__LL_ADC_CALC_DATA_TO_VOLTAGE

Description:

- Helper macro to calculate the voltage (unit: mVolt) corresponding to a ADC conversion data (unit: digital value).

Parameters:

- __VREFANALOG_VOLTAGE__: Analog reference voltage (unit mV)
- __ADC_DATA__: ADC conversion data (resolution 12 bits) (unit: digital value).
- __ADC_RESOLUTION__: This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B

Return value:

- ADC: conversion data equivalent voltage value (unit: mVolt)

Notes:

- Analog reference voltage (Vref+) must be either known from user board environment or can be calculated using ADC measurement and ADC helper macro __LL_ADC_CALC_VREFANALOG_VOLTAGE().

__LL_ADC_CALC_TEMPERATURE

Description:

- Helper macro to calculate the temperature (unit: degree Celsius) from ADC conversion data of internal temperature sensor.

Parameters:

- __VREFANALOG_VOLTAGE__: Analog reference voltage (unit mV)
- __TEMPSENSOR_ADC_DATA__: ADC conversion data of internal temperature sensor (unit: digital value).
- __ADC_RESOLUTION__: ADC resolution at which internal temperature sensor voltage has been measured. This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B

Return value:

- Temperature: (unit: degree Celsius)

Notes:

- Computation is using temperature sensor calibration values stored in system memory for each device during production. Calculation formula: Temperature = ((TS_ADC_DATA - TS_CAL1) * (TS_CAL2_TEMP - TS_CAL1_TEMP)) / (TS_CAL2 - TS_CAL1) + TS_CAL1_TEMP with TS_ADC_DATA = temperature sensor raw data measured by ADC Avg_Slope = (TS_CAL2 - TS_CAL1) / (TS_CAL2_TEMP - TS_CAL1_TEMP) TS_CAL1 = equivalent TS_ADC_DATA at temperature TEMP_DEGC_CAL1 (calibrated in factory) TS_CAL2 = equivalent TS_ADC_DATA at temperature TEMP_DEGC_CAL2 (calibrated in factory) Caution: Calculation relevancy under reserve that calibration parameters are correct (address and data). To calculate temperature using temperature sensor datasheet typical values (generic values less, therefore less accurate than calibrated values), use helper macro __LL_ADC_CALC_TEMPERATURE_TYP_PARAMS(). As calculation input, the analog reference voltage (Vref+) must be defined as it impacts the ADC LSB equivalent voltage. Analog reference voltage (Vref+) must be either known from user board environment or can be calculated using ADC measurement and ADC helper macro __LL_ADC_CALC_VREFANALOG_VOLTAGE(). On this STM32 serie, calibration data of temperature sensor corresponds to a resolution of 12 bits, this is the recommended ADC resolution to convert voltage of temperature sensor. Otherwise, this macro performs the processing to scale ADC conversion data to 12 bits.

__LL_ADC_CALC_TEMPERATURE_TYP_PARAMS

Description:

- Helper macro to calculate the temperature (unit: degree Celsius) from ADC conversion data of internal temperature sensor.

Parameters:

- __TEMPSENSOR_TYP_AVGSLOPE__: Device datasheet data Temperature sensor slope typical value (unit uV/DegCelsius). On STM32F4, refer to device datasheet parameter "Avg_Slope".
- __TEMPSENSOR_TYP_CALX_V__: Device datasheet data Temperature sensor voltage typical value (at temperature and Vref+ defined in parameters below) (unit mV). On STM32F4, refer to device datasheet parameter "V25".
- __TEMPSENSOR_CALX_TEMP__: Device datasheet data Temperature at which temperature sensor voltage (see parameter above) is corresponding (unit mV)
- __VREFANALOG_VOLTAGE__: Analog voltage reference (Vref+) voltage (unit mV)
- __TEMPSENSOR_ADC_DATA__: ADC conversion data of internal temperature sensor (unit digital value).
- __ADC_RESOLUTION__: ADC resolution at which internal temperature sensor voltage has been measured. This parameter can be one of the following values:
  - LL_ADC_RESOLUTION_12B
  - LL_ADC_RESOLUTION_10B
  - LL_ADC_RESOLUTION_8B
  - LL_ADC_RESOLUTION_6B

Return value:

- Temperature: (unit: degree Celsius)

Notes:

- Computation is using temperature sensor typical values (refer to device datasheet). Calculation formula: Temperature = (TS_TYP_CALx_VOLT(uV) - TS_ADC_DATA * Conversion_uV) / Avg_Slope + CALx_TEMP with TS_ADC_DATA = temperature sensor raw data measured by ADC (unit: digital value) Avg_Slope = temperature sensor slope (unit: uV/Degree Celsius) TS_TYP_CALx_VOLT = temperature sensor digital value at temperature CALx_TEMP (unit: mV) Caution: Calculation relevancy under reserve the temperature sensor of the current device has characteristics in line with datasheet typical values. If temperature sensor calibration values are available on on this device (presence of macro __LL_ADC_CALC_TEMPERATURE()), temperature calculation will be more accurate using helper macro __LL_ADC_CALC_TEMPERATURE(). As calculation input, the analog reference voltage (Vref+) must be defined as it impacts the ADC LSB equivalent voltage. Analog reference voltage (Vref+) must be either known from user board environment or can be calculated using ADC measurement and ADC helper macro __LL_ADC_CALC_VREFANALOG_VOLTAGE(). ADC measurement data must correspond to a resolution of 12bits (full scale digital value 4095). If not the case, the data must be preliminarily rescaled to an equivalent resolution of 12 bits.

Common write and read registers Macros

LL_ADC_WriteReg

Description:

- Write a value in ADC register.

Parameters:

- __INSTANCE__: ADC Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_ADC_ReadReg

Description:

- Read a value in ADC register.

Parameters:

- __INSTANCE__: ADC Instance
- __REG__: Register to be read

Return value:

- Register: value

74 LL BUS Generic Driver

74.1 BUS Firmware driver API description

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

74.1.1 Detailed description of functions

LL_AHB1_GRP1_EnableClock

Function name

__STATIC_INLINE void LL_AHB1_GRP1_EnableClock (uint32_t Periphs)

Function description

Enable AHB1 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB1_GRP1_PERIPH_GPIOA
  - LL_AHB1_GRP1_PERIPH_GPIOB
  - LL_AHB1_GRP1_PERIPH_GPIOC
  - LL_AHB1_GRP1_PERIPH_GPIOD (*)
  - LL_AHB1_GRP1_PERIPH_GPIOE (*)
  - LL_AHB1_GRP1_PERIPH_GPIOF (*)
  - LL_AHB1_GRP1_PERIPH_GPIOG (*)
  - LL_AHB1_GRP1_PERIPH_GPIOH (*)
  - LL_AHB1_GRP1_PERIPH_GPIOI (*)
  - LL_AHB1_GRP1_PERIPH_GPIOJ (*)
  - LL_AHB1_GRP1_PERIPH_GPIOK (*)
  - LL_AHB1_GRP1_PERIPH_CRC
  - LL_AHB1_GRP1_PERIPH_BKPSRAM (*)
  - LL_AHB1_GRP1_PERIPH_CCMDATARAM (*)
  - LL_AHB1_GRP1_PERIPH_DMA1
  - LL_AHB1_GRP1_PERIPH_DMA2
  - LL_AHB1_GRP1_PERIPH_RNG (*)
  - LL_AHB1_GRP1_PERIPH_DMA2D (*)
  - LL_AHB1_GRP1_PERIPH_ETHMAC (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACTX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACRX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACPTP (*)
  - LL_AHB1_GRP1_PERIPH_OTGHS (*)
  - LL_AHB1_GRP1_PERIPH_OTGHSULPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB1ENR GPIOAEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOBEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOCEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIODEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOEEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOFEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOGEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOHEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOIEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOJEN LL_AHB1_GRP1_EnableClock
- AHB1ENR GPIOKEN LL_AHB1_GRP1_EnableClock
- AHB1ENR CRCEN LL_AHB1_GRP1_EnableClock
- AHB1ENR BKPSRAMEN LL_AHB1_GRP1_EnableClock
- AHB1ENR CCMDATARAMEN LL_AHB1_GRP1_EnableClock
- AHB1ENR DMA1EN LL_AHB1_GRP1_EnableClock
- AHB1ENR DMA2EN LL_AHB1_GRP1_EnableClock
- AHB1ENR RNGEN LL_AHB1_GRP1_EnableClock
- AHB1ENR DMA2DEN LL_AHB1_GRP1_EnableClock
- AHB1ENR ETHMACEN LL_AHB1_GRP1_EnableClock
- AHB1ENR ETHMACTXEN LL_AHB1_GRP1_EnableClock
- AHB1ENR ETHMACRXEN LL_AHB1_GRP1_EnableClock
- AHB1ENR ETHMACPTPEN LL_AHB1_GRP1_EnableClock
- AHB1ENR OTGHSEN LL_AHB1_GRP1_EnableClock
- AHB1ENR OTGHSULPIEN LL_AHB1_GRP1_EnableClock

LL_AHB1_GRP1_IsEnabledClock

Function name

__STATIC_INLINE uint32_t LL_AHB1_GRP1_IsEnabledClock (uint32_t Periphs)

Function description

Check if AHB1 peripheral clock is enabled or not.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB1_GRP1_PERIPH_GPIOA
  - LL_AHB1_GRP1_PERIPH_GPIOB
  - LL_AHB1_GRP1_PERIPH_GPIOC
  - LL_AHB1_GRP1_PERIPH_GPIOD (*)
  - LL_AHB1_GRP1_PERIPH_GPIOE (*)
  - LL_AHB1_GRP1_PERIPH_GPIOF (*)
  - LL_AHB1_GRP1_PERIPH_GPIOG (*)
  - LL_AHB1_GRP1_PERIPH_GPIOH (*)
  - LL_AHB1_GRP1_PERIPH_GPIOI (*)
  - LL_AHB1_GRP1_PERIPH_GPIOJ (*)
  - LL_AHB1_GRP1_PERIPH_GPIOK (*)
  - LL_AHB1_GRP1_PERIPH_CRC
  - LL_AHB1_GRP1_PERIPH_BKPSRAM (*)
  - LL_AHB1_GRP1_PERIPH_CCMDATARAM (*)
  - LL_AHB1_GRP1_PERIPH_DMA1
  - LL_AHB1_GRP1_PERIPH_DMA2
  - LL_AHB1_GRP1_PERIPH_RNG (*)
  - LL_AHB1_GRP1_PERIPH_DMA2D (*)
  - LL_AHB1_GRP1_PERIPH_ETHMAC (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACTX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACRX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACPTP (*)
  - LL_AHB1_GRP1_PERIPH_OTGHS (*)
  - LL_AHB1_GRP1_PERIPH_OTGHSULPI (*)

  (*) value not defined in all devices.

Return values

- **State:** of Periphs (1 or 0).

Reference Manual to LL API cross reference:

- AHB1ENR GPIOAEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOBEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOCEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIODEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOEEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOFEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOGEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOHEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOIEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOJEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR GPIOKEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR CRCEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR BKPSRAMEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR CCMDATARAMEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR DMA1EN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR DMA2EN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR RNGEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR DMA2DEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR ETHMACEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR ETHMACTXEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR ETHMACRXEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR ETHMACPTPEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR OTGHSEN LL_AHB1_GRP1_IsEnabledClock
- AHB1ENR OTGHSULPIEN LL_AHB1_GRP1_IsEnabledClock

LL_AHB1_GRP1_DisableClock

Function name

__STATIC_INLINE void LL_AHB1_GRP1_DisableClock (uint32_t Periphs)

Function description

Disable AHB1 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB1_GRP1_PERIPH_GPIOA
  - LL_AHB1_GRP1_PERIPH_GPIOB
  - LL_AHB1_GRP1_PERIPH_GPIOC
  - LL_AHB1_GRP1_PERIPH_GPIOD (*)
  - LL_AHB1_GRP1_PERIPH_GPIOE (*)
  - LL_AHB1_GRP1_PERIPH_GPIOF (*)
  - LL_AHB1_GRP1_PERIPH_GPIOG (*)
  - LL_AHB1_GRP1_PERIPH_GPIOH (*)
  - LL_AHB1_GRP1_PERIPH_GPIOI (*)
  - LL_AHB1_GRP1_PERIPH_GPIOJ (*)
  - LL_AHB1_GRP1_PERIPH_GPIOK (*)
  - LL_AHB1_GRP1_PERIPH_CRC
  - LL_AHB1_GRP1_PERIPH_BKPSRAM (*)
  - LL_AHB1_GRP1_PERIPH_CCMDATARAM (*)
  - LL_AHB1_GRP1_PERIPH_DMA1
  - LL_AHB1_GRP1_PERIPH_DMA2
  - LL_AHB1_GRP1_PERIPH_RNG (*)
  - LL_AHB1_GRP1_PERIPH_DMA2D (*)
  - LL_AHB1_GRP1_PERIPH_ETHMAC (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACTX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACRX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACPTP (*)
  - LL_AHB1_GRP1_PERIPH_OTGHS (*)
  - LL_AHB1_GRP1_PERIPH_OTGHSULPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB1ENR GPIOAEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOBEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOCEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIODEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOEEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOFEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOGEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOHEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOIEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOJEN LL_AHB1_GRP1_DisableClock
- AHB1ENR GPIOKEN LL_AHB1_GRP1_DisableClock
- AHB1ENR CRCEN LL_AHB1_GRP1_DisableClock
- AHB1ENR BKPSRAMEN LL_AHB1_GRP1_DisableClock
- AHB1ENR CCMDATARAMEN LL_AHB1_GRP1_DisableClock
- AHB1ENR DMA1EN LL_AHB1_GRP1_DisableClock
- AHB1ENR DMA2EN LL_AHB1_GRP1_DisableClock
- AHB1ENR RNGEN LL_AHB1_GRP1_DisableClock
- AHB1ENR DMA2DEN LL_AHB1_GRP1_DisableClock
- AHB1ENR ETHMACEN LL_AHB1_GRP1_DisableClock
- AHB1ENR ETHMACTXEN LL_AHB1_GRP1_DisableClock
- AHB1ENR ETHMACRXEN LL_AHB1_GRP1_DisableClock
- AHB1ENR ETHMACPTPEN LL_AHB1_GRP1_DisableClock
- AHB1ENR OTGHSEN LL_AHB1_GRP1_DisableClock
- AHB1ENR OTGHSULPIEN LL_AHB1_GRP1_DisableClock

LL_AHB1_GRP1_ForceReset

Function name

__STATIC_INLINE void LL_AHB1_GRP1_ForceReset (uint32_t Periphs)

Function description

Force AHB1 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB1_GRP1_PERIPH_ALL
  - LL_AHB1_GRP1_PERIPH_GPIOA
  - LL_AHB1_GRP1_PERIPH_GPIOB
  - LL_AHB1_GRP1_PERIPH_GPIOC
  - LL_AHB1_GRP1_PERIPH_GPIOD (*)
  - LL_AHB1_GRP1_PERIPH_GPIOE (*)
  - LL_AHB1_GRP1_PERIPH_GPIOF (*)
  - LL_AHB1_GRP1_PERIPH_GPIOG (*)
  - LL_AHB1_GRP1_PERIPH_GPIOH (*)
  - LL_AHB1_GRP1_PERIPH_GPIOI (*)
  - LL_AHB1_GRP1_PERIPH_GPIOJ (*)
  - LL_AHB1_GRP1_PERIPH_GPIOK (*)
  - LL_AHB1_GRP1_PERIPH_CRC
  - LL_AHB1_GRP1_PERIPH_DMA1
  - LL_AHB1_GRP1_PERIPH_DMA2
  - LL_AHB1_GRP1_PERIPH_RNG (*)
  - LL_AHB1_GRP1_PERIPH_DMA2D (*)
  - LL_AHB1_GRP1_PERIPH_ETHMAC (*)
  - LL_AHB1_GRP1_PERIPH_OTGHS (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB1RSTR GPIOARST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOBRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOCRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIODRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOERST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOFRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOGRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOHRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOIRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOJRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR GPIOKRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR CRCRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR DMA1RST LL_AHB1_GRP1_ForceReset
- AHB1RSTR DMA2RST LL_AHB1_GRP1_ForceReset
- AHB1RSTR RNGRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR DMA2DRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR ETHMACRST LL_AHB1_GRP1_ForceReset
- AHB1RSTR OTGHSRST LL_AHB1_GRP1_ForceReset

LL_AHB1_GRP1_ReleaseReset

Function name

__STATIC_INLINE void LL_AHB1_GRP1_ReleaseReset (uint32_t Periphs)

Function description

Release AHB1 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB1_GRP1_PERIPH_ALL
  - LL_AHB1_GRP1_PERIPH_GPIOA
  - LL_AHB1_GRP1_PERIPH_GPIOB
  - LL_AHB1_GRP1_PERIPH_GPIOC
  - LL_AHB1_GRP1_PERIPH_GPIOD (*)
  - LL_AHB1_GRP1_PERIPH_GPIOE (*)
  - LL_AHB1_GRP1_PERIPH_GPIOF (*)
  - LL_AHB1_GRP1_PERIPH_GPIOG (*)
  - LL_AHB1_GRP1_PERIPH_GPIOH (*)
  - LL_AHB1_GRP1_PERIPH_GPIOI (*)
  - LL_AHB1_GRP1_PERIPH_GPIOJ (*)
  - LL_AHB1_GRP1_PERIPH_GPIOK (*)
  - LL_AHB1_GRP1_PERIPH_CRC
  - LL_AHB1_GRP1_PERIPH_DMA1
  - LL_AHB1_GRP1_PERIPH_DMA2
  - LL_AHB1_GRP1_PERIPH_RNG (*)
  - LL_AHB1_GRP1_PERIPH_DMA2D (*)
  - LL_AHB1_GRP1_PERIPH_ETHMAC (*)
  - LL_AHB1_GRP1_PERIPH_OTGHS (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB1RSTR GPIOARST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOBRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOCRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIODRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOERST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOFRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOGRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOHRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOIRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOJRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR GPIOKRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR CRCRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR DMA1RST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR DMA2RST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR RNGRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR DMA2DRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR ETHMACRST LL_AHB1_GRP1_ReleaseReset
- AHB1RSTR OTGHSRST LL_AHB1_GRP1_ReleaseReset

LL_AHB1_GRP1_EnableClockLowPower

Function name

__STATIC_INLINE void LL_AHB1_GRP1_EnableClockLowPower (uint32_t Periphs)

Function description

Enable AHB1 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB1_GRP1_PERIPH_GPIOA
  - LL_AHB1_GRP1_PERIPH_GPIOB
  - LL_AHB1_GRP1_PERIPH_GPIOC
  - LL_AHB1_GRP1_PERIPH_GPIOD (*)
  - LL_AHB1_GRP1_PERIPH_GPIOE (*)
  - LL_AHB1_GRP1_PERIPH_GPIOF (*)
  - LL_AHB1_GRP1_PERIPH_GPIOG (*)
  - LL_AHB1_GRP1_PERIPH_GPIOH (*)
  - LL_AHB1_GRP1_PERIPH_GPIOI (*)
  - LL_AHB1_GRP1_PERIPH_GPIOJ (*)
  - LL_AHB1_GRP1_PERIPH_GPIOK (*)
  - LL_AHB1_GRP1_PERIPH_CRC
  - LL_AHB1_GRP1_PERIPH_BKPSRAM (*)
  - LL_AHB1_GRP1_PERIPH_FLITF
  - LL_AHB1_GRP1_PERIPH_SRAM1
  - LL_AHB1_GRP1_PERIPH_SRAM2 (*)
  - LL_AHB1_GRP1_PERIPH_SRAM3 (*)
  - LL_AHB1_GRP1_PERIPH_DMA1
  - LL_AHB1_GRP1_PERIPH_DMA2
  - LL_AHB1_GRP1_PERIPH_RNG (*)
  - LL_AHB1_GRP1_PERIPH_DMA2D (*)
  - LL_AHB1_GRP1_PERIPH_ETHMAC (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACTX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACRX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACPTP (*)
  - LL_AHB1_GRP1_PERIPH_OTGHS (*)
  - LL_AHB1_GRP1_PERIPH_OTGHSULPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB1LPENR GPIOALPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOBLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOCLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIODLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOELPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOFLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOGLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOHLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOILPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOJLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR GPIOKLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR CRCLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR BKPSRAMLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR FLITFLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR SRAM1LPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR SRAM2LPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR SRAM3LPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR BKPSRAMLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR DMA1LPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR DMA2LPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR DMA2DLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR RNGLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR ETHMACLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR ETHMACTXLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR ETHMACRXLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR ETHMACPTPLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR OTGHSLPEN LL_AHB1_GRP1_EnableClockLowPower
- AHB1LPENR OTGHSULPILPEN LL_AHB1_GRP1_EnableClockLowPower

LL_AHB1_GRP1_DisableClockLowPower

Function name

__STATIC_INLINE void LL_AHB1_GRP1_DisableClockLowPower (uint32_t Periphs)

Function description

Disable AHB1 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB1_GRP1_PERIPH_GPIOA
  - LL_AHB1_GRP1_PERIPH_GPIOB
  - LL_AHB1_GRP1_PERIPH_GPIOC
  - LL_AHB1_GRP1_PERIPH_GPIOD (*)
  - LL_AHB1_GRP1_PERIPH_GPIOE (*)
  - LL_AHB1_GRP1_PERIPH_GPIOF (*)
  - LL_AHB1_GRP1_PERIPH_GPIOG (*)
  - LL_AHB1_GRP1_PERIPH_GPIOH (*)
  - LL_AHB1_GRP1_PERIPH_GPIOI (*)
  - LL_AHB1_GRP1_PERIPH_GPIOJ (*)
  - LL_AHB1_GRP1_PERIPH_GPIOK (*)
  - LL_AHB1_GRP1_PERIPH_CRC
  - LL_AHB1_GRP1_PERIPH_BKPSRAM (*)
  - LL_AHB1_GRP1_PERIPH_FLITF
  - LL_AHB1_GRP1_PERIPH_SRAM1
  - LL_AHB1_GRP1_PERIPH_SRAM2 (*)
  - LL_AHB1_GRP1_PERIPH_SRAM3 (*)
  - LL_AHB1_GRP1_PERIPH_DMA1
  - LL_AHB1_GRP1_PERIPH_DMA2
  - LL_AHB1_GRP1_PERIPH_RNG (*)
  - LL_AHB1_GRP1_PERIPH_DMA2D (*)
  - LL_AHB1_GRP1_PERIPH_ETHMAC (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACTX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACRX (*)
  - LL_AHB1_GRP1_PERIPH_ETHMACPTP (*)
  - LL_AHB1_GRP1_PERIPH_OTGHS (*)
  - LL_AHB1_GRP1_PERIPH_OTGHSULPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB1LPENR GPIOALPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOBLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOCLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIODLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOELPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOFLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOGLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOHLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOILPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOJLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR GPIOKLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR CRCLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR BKPSRAMLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR FLITFLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR SRAM1LPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR SRAM2LPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR SRAM3LPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR BKPSRAMLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR DMA1LPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR DMA2LPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR DMA2DLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR RNGLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR ETHMACLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR ETHMACTXLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR ETHMACRXLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR ETHMACPTPLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR OTGHSLPEN LL_AHB1_GRP1_DisableClockLowPower
- AHB1LPENR OTGHSULPILPEN LL_AHB1_GRP1_DisableClockLowPower

LL_AHB2_GRP1_EnableClock

Function name

__STATIC_INLINE void LL_AHB2_GRP1_EnableClock (uint32_t Periphs)

Function description

Enable AHB2 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB2_GRP1_PERIPH_DCMI (*)
  - LL_AHB2_GRP1_PERIPH_CRYP (*)
  - LL_AHB2_GRP1_PERIPH_AES (*)
  - LL_AHB2_GRP1_PERIPH_HASH (*)
  - LL_AHB2_GRP1_PERIPH_RNG (*)
  - LL_AHB2_GRP1_PERIPH_OTGFS (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB2ENR DCMIEN LL_AHB2_GRP1_EnableClock
- AHB2ENR CRYPEN LL_AHB2_GRP1_EnableClock
- AHB2ENR AESEN LL_AHB2_GRP1_EnableClock
- AHB2ENR HASHEN LL_AHB2_GRP1_EnableClock
- AHB2ENR RNGEN LL_AHB2_GRP1_EnableClock
- AHB2ENR OTGFSEN LL_AHB2_GRP1_EnableClock

LL_AHB2_GRP1_IsEnabledClock

Function name

__STATIC_INLINE uint32_t LL_AHB2_GRP1_IsEnabledClock (uint32_t Periphs)

Function description

Check if AHB2 peripheral clock is enabled or not.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB2_GRP1_PERIPH_DCMI (*)
  - LL_AHB2_GRP1_PERIPH_CRYP (*)
  - LL_AHB2_GRP1_PERIPH_AES (*)
  - LL_AHB2_GRP1_PERIPH_HASH (*)
  - LL_AHB2_GRP1_PERIPH_RNG (*)
  - LL_AHB2_GRP1_PERIPH_OTGFS (*)

  (*) value not defined in all devices.

Return values

- **State:** of Periphs (1 or 0).

Reference Manual to LL API cross reference:

- AHB2ENR DCMIEN LL_AHB2_GRP1_IsEnabledClock
- AHB2ENR CRYPEN LL_AHB2_GRP1_IsEnabledClock
- AHB2ENR AESEN LL_AHB2_GRP1_IsEnabledClock
- AHB2ENR HASHEN LL_AHB2_GRP1_IsEnabledClock
- AHB2ENR RNGEN LL_AHB2_GRP1_IsEnabledClock
- AHB2ENR OTGFSEN LL_AHB2_GRP1_IsEnabledClock

LL_AHB2_GRP1_DisableClock

Function name

__STATIC_INLINE void LL_AHB2_GRP1_DisableClock (uint32_t Periphs)

Function description

Disable AHB2 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB2_GRP1_PERIPH_DCMI (*)
  - LL_AHB2_GRP1_PERIPH_CRYP (*)
  - LL_AHB2_GRP1_PERIPH_AES (*)
  - LL_AHB2_GRP1_PERIPH_HASH (*)
  - LL_AHB2_GRP1_PERIPH_RNG (*)
  - LL_AHB2_GRP1_PERIPH_OTGFS (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB2ENR DCMIEN LL_AHB2_GRP1_DisableClock
- AHB2ENR CRYPEN LL_AHB2_GRP1_DisableClock
- AHB2ENR AESEN LL_AHB2_GRP1_DisableClock
- AHB2ENR HASHEN LL_AHB2_GRP1_DisableClock
- AHB2ENR RNGEN LL_AHB2_GRP1_DisableClock
- AHB2ENR OTGFSEN LL_AHB2_GRP1_DisableClock

LL_AHB2_GRP1_ForceReset

Function name

__STATIC_INLINE void LL_AHB2_GRP1_ForceReset (uint32_t Periphs)

Function description

Force AHB2 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB2_GRP1_PERIPH_ALL
  - LL_AHB2_GRP1_PERIPH_DCMI (*)
  - LL_AHB2_GRP1_PERIPH_CRYP (*)
  - LL_AHB2_GRP1_PERIPH_AES (*)
  - LL_AHB2_GRP1_PERIPH_HASH (*)
  - LL_AHB2_GRP1_PERIPH_RNG (*)
  - LL_AHB2_GRP1_PERIPH_OTGFS (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB2RSTR DCMIRST LL_AHB2_GRP1_ForceReset
- AHB2RSTR CRYPRST LL_AHB2_GRP1_ForceReset
- AHB2RSTR AESRST LL_AHB2_GRP1_ForceReset
- AHB2RSTR HASHRST LL_AHB2_GRP1_ForceReset
- AHB2RSTR RNGRST LL_AHB2_GRP1_ForceReset
- AHB2RSTR OTGFSRST LL_AHB2_GRP1_ForceReset

LL_AHB2_GRP1_ReleaseReset

Function name

__STATIC_INLINE void LL_AHB2_GRP1_ReleaseReset (uint32_t Periphs)

Function description

Release AHB2 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB2_GRP1_PERIPH_ALL
  - LL_AHB2_GRP1_PERIPH_DCMI (*)
  - LL_AHB2_GRP1_PERIPH_CRYP (*)
  - LL_AHB2_GRP1_PERIPH_AES (*)
  - LL_AHB2_GRP1_PERIPH_HASH (*)
  - LL_AHB2_GRP1_PERIPH_RNG (*)
  - LL_AHB2_GRP1_PERIPH_OTGFS (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB2RSTR DCMIRST LL_AHB2_GRP1_ReleaseReset
- AHB2RSTR CRYPRST LL_AHB2_GRP1_ReleaseReset
- AHB2RSTR AESRST LL_AHB2_GRP1_ReleaseReset
- AHB2RSTR HASHRST LL_AHB2_GRP1_ReleaseReset
- AHB2RSTR RNGRST LL_AHB2_GRP1_ReleaseReset
- AHB2RSTR OTGFSRST LL_AHB2_GRP1_ReleaseReset

LL_AHB2_GRP1_EnableClockLowPower

Function name

__STATIC_INLINE void LL_AHB2_GRP1_EnableClockLowPower (uint32_t Periphs)

Function description

Enable AHB2 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB2_GRP1_PERIPH_DCMI (*)
  - LL_AHB2_GRP1_PERIPH_CRYP (*)
  - LL_AHB2_GRP1_PERIPH_AES (*)
  - LL_AHB2_GRP1_PERIPH_HASH (*)
  - LL_AHB2_GRP1_PERIPH_RNG (*)
  - LL_AHB2_GRP1_PERIPH_OTGFS (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB2LPENR DCMILPEN LL_AHB2_GRP1_EnableClockLowPower
- AHB2LPENR CRYPLPEN LL_AHB2_GRP1_EnableClockLowPower
- AHB2LPENR AESLPEN LL_AHB2_GRP1_EnableClockLowPower
- AHB2LPENR HASHLPEN LL_AHB2_GRP1_EnableClockLowPower
- AHB2LPENR RNGLPEN LL_AHB2_GRP1_EnableClockLowPower
- AHB2LPENR OTGFSLPEN LL_AHB2_GRP1_EnableClockLowPower

LL_AHB2_GRP1_DisableClockLowPower

Function name

__STATIC_INLINE void LL_AHB2_GRP1_DisableClockLowPower (uint32_t Periphs)

Function description

Disable AHB2 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB2_GRP1_PERIPH_DCMI (*)
  - LL_AHB2_GRP1_PERIPH_CRYP (*)
  - LL_AHB2_GRP1_PERIPH_AES (*)
  - LL_AHB2_GRP1_PERIPH_HASH (*)
  - LL_AHB2_GRP1_PERIPH_RNG (*)
  - LL_AHB2_GRP1_PERIPH_OTGFS (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB2LPENR DCMILPEN LL_AHB2_GRP1_DisableClockLowPower
- AHB2LPENR CRYPLPEN LL_AHB2_GRP1_DisableClockLowPower
- AHB2LPENR AESLPEN LL_AHB2_GRP1_DisableClockLowPower
- AHB2LPENR HASHLPEN LL_AHB2_GRP1_DisableClockLowPower
- AHB2LPENR RNGLPEN LL_AHB2_GRP1_DisableClockLowPower
- AHB2LPENR OTGFSLPEN LL_AHB2_GRP1_DisableClockLowPower

LL_AHB3_GRP1_EnableClock

Function name

__STATIC_INLINE void LL_AHB3_GRP1_EnableClock (uint32_t Periphs)

Function description

Enable AHB3 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB3_GRP1_PERIPH_FMC (*)
  - LL_AHB3_GRP1_PERIPH_FSMC (*)
  - LL_AHB3_GRP1_PERIPH_QSPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB3ENR FMCEN LL_AHB3_GRP1_EnableClock
- AHB3ENR FSMCEN LL_AHB3_GRP1_EnableClock
- AHB3ENR QSPIEN LL_AHB3_GRP1_EnableClock

LL_AHB3_GRP1_IsEnabledClock

Function name

__STATIC_INLINE uint32_t LL_AHB3_GRP1_IsEnabledClock (uint32_t Periphs)

Function description

Check if AHB3 peripheral clock is enabled or not.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB3_GRP1_PERIPH_FMC (*)
  - LL_AHB3_GRP1_PERIPH_FSMC (*)
  - LL_AHB3_GRP1_PERIPH_QSPI (*)

  (*) value not defined in all devices.

Return values

- **State:** of Periphs (1 or 0).

Reference Manual to LL API cross reference:

- AHB3ENR FMCEN LL_AHB3_GRP1_IsEnabledClock
- AHB3ENR FSMCEN LL_AHB3_GRP1_IsEnabledClock
- AHB3ENR QSPIEN LL_AHB3_GRP1_IsEnabledClock

LL_AHB3_GRP1_DisableClock

Function name

__STATIC_INLINE void LL_AHB3_GRP1_DisableClock (uint32_t Periphs)

Function description

Disable AHB3 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB3_GRP1_PERIPH_FMC (*)
  - LL_AHB3_GRP1_PERIPH_FSMC (*)
  - LL_AHB3_GRP1_PERIPH_QSPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB3ENR FMCEN LL_AHB3_GRP1_DisableClock
- AHB3ENR FSMCEN LL_AHB3_GRP1_DisableClock
- AHB3ENR QSPIEN LL_AHB3_GRP1_DisableClock

LL_AHB3_GRP1_ForceReset

Function name

__STATIC_INLINE void LL_AHB3_GRP1_ForceReset (uint32_t Periphs)

Function description

Force AHB3 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB3_GRP1_PERIPH_ALL
  - LL_AHB3_GRP1_PERIPH_FMC (*)
  - LL_AHB3_GRP1_PERIPH_FSMC (*)
  - LL_AHB3_GRP1_PERIPH_QSPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB3RSTR FMCRST LL_AHB3_GRP1_ForceReset
- AHB3RSTR FSMCRST LL_AHB3_GRP1_ForceReset
- AHB3RSTR QSPIRST LL_AHB3_GRP1_ForceReset

LL_AHB3_GRP1_ReleaseReset

Function name

__STATIC_INLINE void LL_AHB3_GRP1_ReleaseReset (uint32_t Periphs)

Function description

Release AHB3 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB2_GRP1_PERIPH_ALL
  - LL_AHB3_GRP1_PERIPH_FMC (*)
  - LL_AHB3_GRP1_PERIPH_FSMC (*)
  - LL_AHB3_GRP1_PERIPH_QSPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB3RSTR FMCRST LL_AHB3_GRP1_ReleaseReset
- AHB3RSTR FSMCRST LL_AHB3_GRP1_ReleaseReset
- AHB3RSTR QSPIRST LL_AHB3_GRP1_ReleaseReset

LL_AHB3_GRP1_EnableClockLowPower

Function name

__STATIC_INLINE void LL_AHB3_GRP1_EnableClockLowPower (uint32_t Periphs)

Function description

Enable AHB3 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB3_GRP1_PERIPH_FMC (*)
  - LL_AHB3_GRP1_PERIPH_FSMC (*)
  - LL_AHB3_GRP1_PERIPH_QSPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB3LPENR FMCLPEN LL_AHB3_GRP1_EnableClockLowPower
- AHB3LPENR FSMCLPEN LL_AHB3_GRP1_EnableClockLowPower
- AHB3LPENR QSPILPEN LL_AHB3_GRP1_EnableClockLowPower

LL_AHB3_GRP1_DisableClockLowPower

Function name

__STATIC_INLINE void LL_AHB3_GRP1_DisableClockLowPower (uint32_t Periphs)

Function description

Disable AHB3 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_AHB3_GRP1_PERIPH_FMC (*)
  - LL_AHB3_GRP1_PERIPH_FSMC (*)
  - LL_AHB3_GRP1_PERIPH_QSPI (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- AHB3LPENR FMCLPEN LL_AHB3_GRP1_DisableClockLowPower
- AHB3LPENR FSMCLPEN LL_AHB3_GRP1_DisableClockLowPower
- AHB3LPENR QSPILPEN LL_AHB3_GRP1_DisableClockLowPower

LL_APB1_GRP1_EnableClock

Function name

__STATIC_INLINE void LL_APB1_GRP1_EnableClock (uint32_t Periphs)

Function description

Enable APB1 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB1_GRP1_PERIPH_TIM2 (*)
  - LL_APB1_GRP1_PERIPH_TIM3 (*)
  - LL_APB1_GRP1_PERIPH_TIM4 (*)
  - LL_APB1_GRP1_PERIPH_TIM5
  - LL_APB1_GRP1_PERIPH_TIM6 (*)
  - LL_APB1_GRP1_PERIPH_TIM7 (*)
  - LL_APB1_GRP1_PERIPH_TIM12 (*)
  - LL_APB1_GRP1_PERIPH_TIM13 (*)
  - LL_APB1_GRP1_PERIPH_TIM14 (*)
  - LL_APB1_GRP1_PERIPH_LPTIM1 (*)
  - LL_APB1_GRP1_PERIPH_WWDG
  - LL_APB1_GRP1_PERIPH_SPI2 (*)
  - LL_APB1_GRP1_PERIPH_SPI3 (*)
  - LL_APB1_GRP1_PERIPH_SPDIFRX (*)
  - LL_APB1_GRP1_PERIPH_USART2
  - LL_APB1_GRP1_PERIPH_USART3 (*)
  - LL_APB1_GRP1_PERIPH_UART4 (*)
  - LL_APB1_GRP1_PERIPH_UART5 (*)
  - LL_APB1_GRP1_PERIPH_I2C1
  - LL_APB1_GRP1_PERIPH_I2C2
  - LL_APB1_GRP1_PERIPH_I2C3 (*)
  - LL_APB1_GRP1_PERIPH_FMPI2C1 (*)
  - LL_APB1_GRP1_PERIPH_CAN1 (*)
  - LL_APB1_GRP1_PERIPH_CAN2 (*)
  - LL_APB1_GRP1_PERIPH_CAN3 (*)
  - LL_APB1_GRP1_PERIPH_CEC (*)
  - LL_APB1_GRP1_PERIPH_PWR
  - LL_APB1_GRP1_PERIPH_DAC1 (*)
  - LL_APB1_GRP1_PERIPH_UART7 (*)
  - LL_APB1_GRP1_PERIPH_UART8 (*)
  - LL_APB1_GRP1_PERIPH_RTCAPB (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB1ENR TIM2EN LL_APB1_GRP1_EnableClock
- APB1ENR TIM3EN LL_APB1_GRP1_EnableClock
- APB1ENR TIM4EN LL_APB1_GRP1_EnableClock
- APB1ENR TIM5EN LL_APB1_GRP1_EnableClock
- APB1ENR TIM6EN LL_APB1_GRP1_EnableClock
- APB1ENR TIM7EN LL_APB1_GRP1_EnableClock
- APB1ENR TIM12EN LL_APB1_GRP1_EnableClock
- APB1ENR TIM13EN LL_APB1_GRP1_EnableClock
- APB1ENR TIM14EN LL_APB1_GRP1_EnableClock
- APB1ENR LPTIM1EN LL_APB1_GRP1_EnableClock
- APB1ENR WWDGEN LL_APB1_GRP1_EnableClock
- APB1ENR SPI2EN LL_APB1_GRP1_EnableClock
- APB1ENR SPI3EN LL_APB1_GRP1_EnableClock
- APB1ENR SPDIFRXEN LL_APB1_GRP1_EnableClock
- APB1ENR USART2EN LL_APB1_GRP1_EnableClock
- APB1ENR USART3EN LL_APB1_GRP1_EnableClock
- APB1ENR UART4EN LL_APB1_GRP1_EnableClock
- APB1ENR UART5EN LL_APB1_GRP1_EnableClock
- APB1ENR I2C1EN LL_APB1_GRP1_EnableClock
- APB1ENR I2C2EN LL_APB1_GRP1_EnableClock
- APB1ENR I2C3EN LL_APB1_GRP1_EnableClock
- APB1ENR FMPI2C1EN LL_APB1_GRP1_EnableClock
- APB1ENR CAN1EN LL_APB1_GRP1_EnableClock
- APB1ENR CAN2EN LL_APB1_GRP1_EnableClock
- APB1ENR CAN3EN LL_APB1_GRP1_EnableClock
- APB1ENR CECEN LL_APB1_GRP1_EnableClock
- APB1ENR PWREN LL_APB1_GRP1_EnableClock
- APB1ENR DACEN LL_APB1_GRP1_EnableClock
- APB1ENR UART7EN LL_APB1_GRP1_EnableClock
- APB1ENR UART8EN LL_APB1_GRP1_EnableClock
- APB1ENR RTCAPBEN LL_APB1_GRP1_EnableClock

LL_APB1_GRP1_IsEnabledClock

Function name

__STATIC_INLINE uint32_t LL_APB1_GRP1_IsEnabledClock (uint32_t Periphs)

Function description

Check if APB1 peripheral clock is enabled or not.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB1_GRP1_PERIPH_TIM2 (*)
  - LL_APB1_GRP1_PERIPH_TIM3 (*)
  - LL_APB1_GRP1_PERIPH_TIM4 (*)
  - LL_APB1_GRP1_PERIPH_TIM5
  - LL_APB1_GRP1_PERIPH_TIM6 (*)
  - LL_APB1_GRP1_PERIPH_TIM7 (*)
  - LL_APB1_GRP1_PERIPH_TIM12 (*)
  - LL_APB1_GRP1_PERIPH_TIM13 (*)
  - LL_APB1_GRP1_PERIPH_TIM14 (*)
  - LL_APB1_GRP1_PERIPH_LPTIM1 (*)
  - LL_APB1_GRP1_PERIPH_WWDG
  - LL_APB1_GRP1_PERIPH_SPI2 (*)
  - LL_APB1_GRP1_PERIPH_SPI3 (*)
  - LL_APB1_GRP1_PERIPH_SPDIFRX (*)
  - LL_APB1_GRP1_PERIPH_USART2
  - LL_APB1_GRP1_PERIPH_USART3 (*)
  - LL_APB1_GRP1_PERIPH_UART4 (*)
  - LL_APB1_GRP1_PERIPH_UART5 (*)
  - LL_APB1_GRP1_PERIPH_I2C1
  - LL_APB1_GRP1_PERIPH_I2C2
  - LL_APB1_GRP1_PERIPH_I2C3 (*)
  - LL_APB1_GRP1_PERIPH_FMPI2C1 (*)
  - LL_APB1_GRP1_PERIPH_CAN1 (*)
  - LL_APB1_GRP1_PERIPH_CAN2 (*)
  - LL_APB1_GRP1_PERIPH_CAN3 (*)
  - LL_APB1_GRP1_PERIPH_CEC (*)
  - LL_APB1_GRP1_PERIPH_PWR
  - LL_APB1_GRP1_PERIPH_DAC1 (*)
  - LL_APB1_GRP1_PERIPH_UART7 (*)
  - LL_APB1_GRP1_PERIPH_UART8 (*)
  - LL_APB1_GRP1_PERIPH_RTCAPB (*)

  (*) value not defined in all devices.

Return values

- **State:** of Periphs (1 or 0).

Reference Manual to LL API cross reference:

- APB1ENR TIM2EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR TIM3EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR TIM4EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR TIM5EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR TIM6EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR TIM7EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR TIM12EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR TIM13EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR TIM14EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR LPTIM1EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR WWDGEN LL_APB1_GRP1_IsEnabledClock
- APB1ENR SPI2EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR SPI3EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR SPDIFRXEN LL_APB1_GRP1_IsEnabledClock
- APB1ENR USART2EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR USART3EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR UART4EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR UART5EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR I2C1EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR I2C2EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR I2C3EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR FMPI2C1EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR CAN1EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR CAN2EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR CAN3EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR CECEN LL_APB1_GRP1_IsEnabledClock
- APB1ENR PWREN LL_APB1_GRP1_IsEnabledClock
- APB1ENR DACEN LL_APB1_GRP1_IsEnabledClock
- APB1ENR UART7EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR UART8EN LL_APB1_GRP1_IsEnabledClock
- APB1ENR RTCAPBEN LL_APB1_GRP1_IsEnabledClock

LL_APB1_GRP1_DisableClock

Function name

__STATIC_INLINE void LL_APB1_GRP1_DisableClock (uint32_t Periphs)

Function description

Disable APB1 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB1_GRP1_PERIPH_TIM2 (*)
  - LL_APB1_GRP1_PERIPH_TIM3 (*)
  - LL_APB1_GRP1_PERIPH_TIM4 (*)
  - LL_APB1_GRP1_PERIPH_TIM5
  - LL_APB1_GRP1_PERIPH_TIM6 (*)
  - LL_APB1_GRP1_PERIPH_TIM7 (*)
  - LL_APB1_GRP1_PERIPH_TIM12 (*)
  - LL_APB1_GRP1_PERIPH_TIM13 (*)
  - LL_APB1_GRP1_PERIPH_TIM14 (*)
  - LL_APB1_GRP1_PERIPH_LPTIM1 (*)
  - LL_APB1_GRP1_PERIPH_WWDG
  - LL_APB1_GRP1_PERIPH_SPI2 (*)
  - LL_APB1_GRP1_PERIPH_SPI3 (*)
  - LL_APB1_GRP1_PERIPH_SPDIFRX (*)
  - LL_APB1_GRP1_PERIPH_USART2
  - LL_APB1_GRP1_PERIPH_USART3 (*)
  - LL_APB1_GRP1_PERIPH_UART4 (*)
  - LL_APB1_GRP1_PERIPH_UART5 (*)
  - LL_APB1_GRP1_PERIPH_I2C1
  - LL_APB1_GRP1_PERIPH_I2C2
  - LL_APB1_GRP1_PERIPH_I2C3 (*)
  - LL_APB1_GRP1_PERIPH_FMPI2C1 (*)
  - LL_APB1_GRP1_PERIPH_CAN1 (*)
  - LL_APB1_GRP1_PERIPH_CAN2 (*)
  - LL_APB1_GRP1_PERIPH_CAN3 (*)
  - LL_APB1_GRP1_PERIPH_CEC (*)
  - LL_APB1_GRP1_PERIPH_PWR
  - LL_APB1_GRP1_PERIPH_DAC1 (*)
  - LL_APB1_GRP1_PERIPH_UART7 (*)
  - LL_APB1_GRP1_PERIPH_UART8 (*)
  - LL_APB1_GRP1_PERIPH_RTCAPB (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB1ENR TIM2EN LL_APB1_GRP1_DisableClock
- APB1ENR TIM3EN LL_APB1_GRP1_DisableClock
- APB1ENR TIM4EN LL_APB1_GRP1_DisableClock
- APB1ENR TIM5EN LL_APB1_GRP1_DisableClock
- APB1ENR TIM6EN LL_APB1_GRP1_DisableClock
- APB1ENR TIM7EN LL_APB1_GRP1_DisableClock
- APB1ENR TIM12EN LL_APB1_GRP1_DisableClock
- APB1ENR TIM13EN LL_APB1_GRP1_DisableClock
- APB1ENR TIM14EN LL_APB1_GRP1_DisableClock
- APB1ENR LPTIM1EN LL_APB1_GRP1_DisableClock
- APB1ENR WWDGEN LL_APB1_GRP1_DisableClock
- APB1ENR SPI2EN LL_APB1_GRP1_DisableClock
- APB1ENR SPI3EN LL_APB1_GRP1_DisableClock
- APB1ENR SPDIFRXEN LL_APB1_GRP1_DisableClock
- APB1ENR USART2EN LL_APB1_GRP1_DisableClock
- APB1ENR USART3EN LL_APB1_GRP1_DisableClock
- APB1ENR UART4EN LL_APB1_GRP1_DisableClock
- APB1ENR UART5EN LL_APB1_GRP1_DisableClock
- APB1ENR I2C1EN LL_APB1_GRP1_DisableClock
- APB1ENR I2C2EN LL_APB1_GRP1_DisableClock
- APB1ENR I2C3EN LL_APB1_GRP1_DisableClock
- APB1ENR FMPI2C1EN LL_APB1_GRP1_DisableClock
- APB1ENR CAN1EN LL_APB1_GRP1_DisableClock
- APB1ENR CAN2EN LL_APB1_GRP1_DisableClock
- APB1ENR CAN3EN LL_APB1_GRP1_DisableClock
- APB1ENR CECEN LL_APB1_GRP1_DisableClock
- APB1ENR PWREN LL_APB1_GRP1_DisableClock
- APB1ENR DACEN LL_APB1_GRP1_DisableClock
- APB1ENR UART7EN LL_APB1_GRP1_DisableClock
- APB1ENR UART8EN LL_APB1_GRP1_DisableClock
- APB1ENR RTCAPBEN LL_APB1_GRP1_DisableClock

LL_APB1_GRP1_ForceReset

Function name

__STATIC_INLINE void LL_APB1_GRP1_ForceReset (uint32_t Periphs)

Function description

Force APB1 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB1_GRP1_PERIPH_TIM2 (*)
  - LL_APB1_GRP1_PERIPH_TIM3 (*)
  - LL_APB1_GRP1_PERIPH_TIM4 (*)
  - LL_APB1_GRP1_PERIPH_TIM5
  - LL_APB1_GRP1_PERIPH_TIM6 (*)
  - LL_APB1_GRP1_PERIPH_TIM7 (*)
  - LL_APB1_GRP1_PERIPH_TIM12 (*)
  - LL_APB1_GRP1_PERIPH_TIM13 (*)
  - LL_APB1_GRP1_PERIPH_TIM14 (*)
  - LL_APB1_GRP1_PERIPH_LPTIM1 (*)
  - LL_APB1_GRP1_PERIPH_WWDG
  - LL_APB1_GRP1_PERIPH_SPI2 (*)
  - LL_APB1_GRP1_PERIPH_SPI3 (*)
  - LL_APB1_GRP1_PERIPH_SPDIFRX (*)
  - LL_APB1_GRP1_PERIPH_USART2
  - LL_APB1_GRP1_PERIPH_USART3 (*)
  - LL_APB1_GRP1_PERIPH_UART4 (*)
  - LL_APB1_GRP1_PERIPH_UART5 (*)
  - LL_APB1_GRP1_PERIPH_I2C1
  - LL_APB1_GRP1_PERIPH_I2C2
  - LL_APB1_GRP1_PERIPH_I2C3 (*)
  - LL_APB1_GRP1_PERIPH_FMPI2C1 (*)
  - LL_APB1_GRP1_PERIPH_CAN1 (*)
  - LL_APB1_GRP1_PERIPH_CAN2 (*)
  - LL_APB1_GRP1_PERIPH_CAN3 (*)
  - LL_APB1_GRP1_PERIPH_CEC (*)
  - LL_APB1_GRP1_PERIPH_PWR
  - LL_APB1_GRP1_PERIPH_DAC1 (*)
  - LL_APB1_GRP1_PERIPH_UART7 (*)
  - LL_APB1_GRP1_PERIPH_UART8 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB1RSTR TIM2RST LL_APB1_GRP1_ForceReset
- APB1RSTR TIM3RST LL_APB1_GRP1_ForceReset
- APB1RSTR TIM4RST LL_APB1_GRP1_ForceReset
- APB1RSTR TIM5RST LL_APB1_GRP1_ForceReset
- APB1RSTR TIM6RST LL_APB1_GRP1_ForceReset
- APB1RSTR TIM7RST LL_APB1_GRP1_ForceReset
- APB1RSTR TIM12RST LL_APB1_GRP1_ForceReset
- APB1RSTR TIM13RST LL_APB1_GRP1_ForceReset
- APB1RSTR TIM14RST LL_APB1_GRP1_ForceReset
- APB1RSTR LPTIM1RST LL_APB1_GRP1_ForceReset
- APB1RSTR WWDGRST LL_APB1_GRP1_ForceReset
- APB1RSTR SPI2RST LL_APB1_GRP1_ForceReset
- APB1RSTR SPI3RST LL_APB1_GRP1_ForceReset
- APB1RSTR SPDIFRXRST LL_APB1_GRP1_ForceReset
- APB1RSTR USART2RST LL_APB1_GRP1_ForceReset
- APB1RSTR USART3RST LL_APB1_GRP1_ForceReset
- APB1RSTR UART4RST LL_APB1_GRP1_ForceReset
- APB1RSTR UART5RST LL_APB1_GRP1_ForceReset
- APB1RSTR I2C1RST LL_APB1_GRP1_ForceReset
- APB1RSTR I2C2RST LL_APB1_GRP1_ForceReset
- APB1RSTR I2C3RST LL_APB1_GRP1_ForceReset
- APB1RSTR FMPI2C1RST LL_APB1_GRP1_ForceReset
- APB1RSTR CAN1RST LL_APB1_GRP1_ForceReset
- APB1RSTR CAN2RST LL_APB1_GRP1_ForceReset
- APB1RSTR CAN3RST LL_APB1_GRP1_ForceReset
- APB1RSTR CECRST LL_APB1_GRP1_ForceReset
- APB1RSTR PWRRST LL_APB1_GRP1_ForceReset
- APB1RSTR DACRST LL_APB1_GRP1_ForceReset
- APB1RSTR UART7RST LL_APB1_GRP1_ForceReset
- APB1RSTR UART8RST LL_APB1_GRP1_ForceReset

LL_APB1_GRP1_ReleaseReset

Function name

__STATIC_INLINE void LL_APB1_GRP1_ReleaseReset (uint32_t Periphs)

Function description

Release APB1 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB1_GRP1_PERIPH_TIM2 (*)
  - LL_APB1_GRP1_PERIPH_TIM3 (*)
  - LL_APB1_GRP1_PERIPH_TIM4 (*)
  - LL_APB1_GRP1_PERIPH_TIM5
  - LL_APB1_GRP1_PERIPH_TIM6 (*)
  - LL_APB1_GRP1_PERIPH_TIM7 (*)
  - LL_APB1_GRP1_PERIPH_TIM12 (*)
  - LL_APB1_GRP1_PERIPH_TIM13 (*)
  - LL_APB1_GRP1_PERIPH_TIM14 (*)
  - LL_APB1_GRP1_PERIPH_LPTIM1 (*)
  - LL_APB1_GRP1_PERIPH_WWDG
  - LL_APB1_GRP1_PERIPH_SPI2 (*)
  - LL_APB1_GRP1_PERIPH_SPI3 (*)
  - LL_APB1_GRP1_PERIPH_SPDIFRX (*)
  - LL_APB1_GRP1_PERIPH_USART2
  - LL_APB1_GRP1_PERIPH_USART3 (*)
  - LL_APB1_GRP1_PERIPH_UART4 (*)
  - LL_APB1_GRP1_PERIPH_UART5 (*)
  - LL_APB1_GRP1_PERIPH_I2C1
  - LL_APB1_GRP1_PERIPH_I2C2
  - LL_APB1_GRP1_PERIPH_I2C3 (*)
  - LL_APB1_GRP1_PERIPH_FMPI2C1 (*)
  - LL_APB1_GRP1_PERIPH_CAN1 (*)
  - LL_APB1_GRP1_PERIPH_CAN2 (*)
  - LL_APB1_GRP1_PERIPH_CAN3 (*)
  - LL_APB1_GRP1_PERIPH_CEC (*)
  - LL_APB1_GRP1_PERIPH_PWR
  - LL_APB1_GRP1_PERIPH_DAC1 (*)
  - LL_APB1_GRP1_PERIPH_UART7 (*)
  - LL_APB1_GRP1_PERIPH_UART8 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB1RSTR TIM2RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR TIM3RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR TIM4RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR TIM5RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR TIM6RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR TIM7RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR TIM12RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR TIM13RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR TIM14RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR LPTIM1RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR WWDGRST LL_APB1_GRP1_ReleaseReset
- APB1RSTR SPI2RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR SPI3RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR SPDIFRXRST LL_APB1_GRP1_ReleaseReset
- APB1RSTR USART2RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR USART3RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR UART4RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR UART5RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR I2C1RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR I2C2RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR I2C3RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR FMPI2C1RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR CAN1RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR CAN2RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR CAN3RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR CECRST LL_APB1_GRP1_ReleaseReset
- APB1RSTR PWRRST LL_APB1_GRP1_ReleaseReset
- APB1RSTR DACRST LL_APB1_GRP1_ReleaseReset
- APB1RSTR UART7RST LL_APB1_GRP1_ReleaseReset
- APB1RSTR UART8RST LL_APB1_GRP1_ReleaseReset

LL_APB1_GRP1_EnableClockLowPower

Function name

__STATIC_INLINE void LL_APB1_GRP1_EnableClockLowPower (uint32_t Periphs)

Function description

Enable APB1 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB1_GRP1_PERIPH_TIM2 (*)
  - LL_APB1_GRP1_PERIPH_TIM3 (*)
  - LL_APB1_GRP1_PERIPH_TIM4 (*)
  - LL_APB1_GRP1_PERIPH_TIM5
  - LL_APB1_GRP1_PERIPH_TIM6 (*)
  - LL_APB1_GRP1_PERIPH_TIM7 (*)
  - LL_APB1_GRP1_PERIPH_TIM12 (*)
  - LL_APB1_GRP1_PERIPH_TIM13 (*)
  - LL_APB1_GRP1_PERIPH_TIM14 (*)
  - LL_APB1_GRP1_PERIPH_LPTIM1 (*)
  - LL_APB1_GRP1_PERIPH_WWDG
  - LL_APB1_GRP1_PERIPH_SPI2 (*)
  - LL_APB1_GRP1_PERIPH_SPI3 (*)
  - LL_APB1_GRP1_PERIPH_SPDIFRX (*)
  - LL_APB1_GRP1_PERIPH_USART2
  - LL_APB1_GRP1_PERIPH_USART3 (*)
  - LL_APB1_GRP1_PERIPH_UART4 (*)
  - LL_APB1_GRP1_PERIPH_UART5 (*)
  - LL_APB1_GRP1_PERIPH_I2C1
  - LL_APB1_GRP1_PERIPH_I2C2
  - LL_APB1_GRP1_PERIPH_I2C3 (*)
  - LL_APB1_GRP1_PERIPH_FMPI2C1 (*)
  - LL_APB1_GRP1_PERIPH_CAN1 (*)
  - LL_APB1_GRP1_PERIPH_CAN2 (*)
  - LL_APB1_GRP1_PERIPH_CAN3 (*)
  - LL_APB1_GRP1_PERIPH_CEC (*)
  - LL_APB1_GRP1_PERIPH_PWR
  - LL_APB1_GRP1_PERIPH_DAC1 (*)
  - LL_APB1_GRP1_PERIPH_UART7 (*)
  - LL_APB1_GRP1_PERIPH_UART8 (*)
  - LL_APB1_GRP1_PERIPH_RTCAPB (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB1LPENR TIM2LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR TIM3LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR TIM4LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR TIM5LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR TIM6LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR TIM7LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR TIM12LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR TIM13LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR TIM14LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR LPTIM1LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR WWDGLPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR SPI2LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR SPI3LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR SPDIFRXLPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR USART2LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR USART3LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR UART4LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR UART5LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR I2C1LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR I2C2LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR I2C3LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR FMPI2C1LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR CAN1LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR CAN2LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR CAN3LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR CECLPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR PWRLPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR DACLPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR UART7LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR UART8LPEN LL_APB1_GRP1_EnableClockLowPower
- APB1LPENR RTCAPBLPEN LL_APB1_GRP1_EnableClockLowPower

LL_APB1_GRP1_DisableClockLowPower

Function name

__STATIC_INLINE void LL_APB1_GRP1_DisableClockLowPower (uint32_t Periphs)

Function description

Disable APB1 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB1_GRP1_PERIPH_TIM2 (*)
  - LL_APB1_GRP1_PERIPH_TIM3 (*)
  - LL_APB1_GRP1_PERIPH_TIM4 (*)
  - LL_APB1_GRP1_PERIPH_TIM5
  - LL_APB1_GRP1_PERIPH_TIM6 (*)
  - LL_APB1_GRP1_PERIPH_TIM7 (*)
  - LL_APB1_GRP1_PERIPH_TIM12 (*)
  - LL_APB1_GRP1_PERIPH_TIM13 (*)
  - LL_APB1_GRP1_PERIPH_TIM14 (*)
  - LL_APB1_GRP1_PERIPH_LPTIM1 (*)
  - LL_APB1_GRP1_PERIPH_WWDG
  - LL_APB1_GRP1_PERIPH_SPI2 (*)
  - LL_APB1_GRP1_PERIPH_SPI3 (*)
  - LL_APB1_GRP1_PERIPH_SPDIFRX (*)
  - LL_APB1_GRP1_PERIPH_USART2
  - LL_APB1_GRP1_PERIPH_USART3 (*)
  - LL_APB1_GRP1_PERIPH_UART4 (*)
  - LL_APB1_GRP1_PERIPH_UART5 (*)
  - LL_APB1_GRP1_PERIPH_I2C1
  - LL_APB1_GRP1_PERIPH_I2C2
  - LL_APB1_GRP1_PERIPH_I2C3 (*)
  - LL_APB1_GRP1_PERIPH_FMPI2C1 (*)
  - LL_APB1_GRP1_PERIPH_CAN1 (*)
  - LL_APB1_GRP1_PERIPH_CAN2 (*)
  - LL_APB1_GRP1_PERIPH_CAN3 (*)
  - LL_APB1_GRP1_PERIPH_CEC (*)
  - LL_APB1_GRP1_PERIPH_PWR
  - LL_APB1_GRP1_PERIPH_DAC1 (*)
  - LL_APB1_GRP1_PERIPH_UART7 (*)
  - LL_APB1_GRP1_PERIPH_UART8 (*)
  - LL_APB1_GRP1_PERIPH_RTCAPB (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB1LPENR TIM2LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR TIM3LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR TIM4LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR TIM5LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR TIM6LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR TIM7LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR TIM12LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR TIM13LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR TIM14LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR LPTIM1LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR WWDGLPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR SPI2LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR SPI3LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR SPDIFRXLPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR USART2LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR USART3LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR UART4LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR UART5LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR I2C1LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR I2C2LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR I2C3LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR FMPI2C1LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR CAN1LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR CAN2LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR CAN3LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR CECLPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR PWRLPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR DACLPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR UART7LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR UART8LPEN LL_APB1_GRP1_DisableClockLowPower
- APB1LPENR RTCAPBLPEN LL_APB1_GRP1_DisableClockLowPower

LL_APB2_GRP1_EnableClock

Function name

__STATIC_INLINE void LL_APB2_GRP1_EnableClock (uint32_t Periphs)

Function description

Enable APB2 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB2_GRP1_PERIPH_TIM1
  - LL_APB2_GRP1_PERIPH_TIM8 (*)
  - LL_APB2_GRP1_PERIPH_USART1
  - LL_APB2_GRP1_PERIPH_USART6 (*)
  - LL_APB2_GRP1_PERIPH_UART9 (*)
  - LL_APB2_GRP1_PERIPH_UART10 (*)
  - LL_APB2_GRP1_PERIPH_ADC1
  - LL_APB2_GRP1_PERIPH_ADC2 (*)
  - LL_APB2_GRP1_PERIPH_ADC3 (*)
  - LL_APB2_GRP1_PERIPH_SDIO (*)
  - LL_APB2_GRP1_PERIPH_SPI1
  - LL_APB2_GRP1_PERIPH_SPI4 (*)
  - LL_APB2_GRP1_PERIPH_SYSCFG
  - LL_APB2_GRP1_PERIPH_EXTI (*)
  - LL_APB2_GRP1_PERIPH_TIM9
  - LL_APB2_GRP1_PERIPH_TIM10 (*)
  - LL_APB2_GRP1_PERIPH_TIM11
  - LL_APB2_GRP1_PERIPH_SPI5 (*)
  - LL_APB2_GRP1_PERIPH_SPI6 (*)
  - LL_APB2_GRP1_PERIPH_SAI1 (*)
  - LL_APB2_GRP1_PERIPH_SAI2 (*)
  - LL_APB2_GRP1_PERIPH_LTDC (*)
  - LL_APB2_GRP1_PERIPH_DSI (*)
  - LL_APB2_GRP1_PERIPH_DFSDM1 (*)
  - LL_APB2_GRP1_PERIPH_DFSDM2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB2ENR TIM1EN LL_APB2_GRP1_EnableClock
- APB2ENR TIM8EN LL_APB2_GRP1_EnableClock
- APB2ENR USART1EN LL_APB2_GRP1_EnableClock
- APB2ENR USART6EN LL_APB2_GRP1_EnableClock
- APB2ENR UART9EN LL_APB2_GRP1_EnableClock
- APB2ENR UART10EN LL_APB2_GRP1_EnableClock
- APB2ENR ADC1EN LL_APB2_GRP1_EnableClock
- APB2ENR ADC2EN LL_APB2_GRP1_EnableClock
- APB2ENR ADC3EN LL_APB2_GRP1_EnableClock
- APB2ENR SDIOEN LL_APB2_GRP1_EnableClock
- APB2ENR SPI1EN LL_APB2_GRP1_EnableClock
- APB2ENR SPI4EN LL_APB2_GRP1_EnableClock
- APB2ENR SYSCFGEN LL_APB2_GRP1_EnableClock
- APB2ENR EXTITEN LL_APB2_GRP1_EnableClock
- APB2ENR TIM9EN LL_APB2_GRP1_EnableClock
- APB2ENR TIM10EN LL_APB2_GRP1_EnableClock
- APB2ENR TIM11EN LL_APB2_GRP1_EnableClock
- APB2ENR SPI5EN LL_APB2_GRP1_EnableClock
- APB2ENR SPI6EN LL_APB2_GRP1_EnableClock
- APB2ENR SAI1EN LL_APB2_GRP1_EnableClock
- APB2ENR SAI2EN LL_APB2_GRP1_EnableClock
- APB2ENR LTDCEN LL_APB2_GRP1_EnableClock
- APB2ENR DSIEN LL_APB2_GRP1_EnableClock
- APB2ENR DFSDM1EN LL_APB2_GRP1_EnableClock
- APB2ENR DFSDM2EN LL_APB2_GRP1_EnableClock

LL_APB2_GRP1_IsEnabledClock

Function name

__STATIC_INLINE uint32_t LL_APB2_GRP1_IsEnabledClock (uint32_t Periphs)

Function description

Check if APB2 peripheral clock is enabled or not.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB2_GRP1_PERIPH_TIM1
  - LL_APB2_GRP1_PERIPH_TIM8 (*)
  - LL_APB2_GRP1_PERIPH_USART1
  - LL_APB2_GRP1_PERIPH_USART6 (*)
  - LL_APB2_GRP1_PERIPH_UART9 (*)
  - LL_APB2_GRP1_PERIPH_UART10 (*)
  - LL_APB2_GRP1_PERIPH_ADC1
  - LL_APB2_GRP1_PERIPH_ADC2 (*)
  - LL_APB2_GRP1_PERIPH_ADC3 (*)
  - LL_APB2_GRP1_PERIPH_SDIO (*)
  - LL_APB2_GRP1_PERIPH_SPI1
  - LL_APB2_GRP1_PERIPH_SPI4 (*)
  - LL_APB2_GRP1_PERIPH_SYSCFG
  - LL_APB2_GRP1_PERIPH_EXTI (*)
  - LL_APB2_GRP1_PERIPH_TIM9
  - LL_APB2_GRP1_PERIPH_TIM10 (*)
  - LL_APB2_GRP1_PERIPH_TIM11
  - LL_APB2_GRP1_PERIPH_SPI5 (*)
  - LL_APB2_GRP1_PERIPH_SPI6 (*)
  - LL_APB2_GRP1_PERIPH_SAI1 (*)
  - LL_APB2_GRP1_PERIPH_SAI2 (*)
  - LL_APB2_GRP1_PERIPH_LTDC (*)
  - LL_APB2_GRP1_PERIPH_DSI (*)
  - LL_APB2_GRP1_PERIPH_DFSDM1 (*)
  - LL_APB2_GRP1_PERIPH_DFSDM2 (*)

  (*) value not defined in all devices.

Return values

- **State:** of Periphs (1 or 0).

Reference Manual to LL API cross reference:

- APB2ENR TIM1EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR TIM8EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR USART1EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR USART6EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR UART9EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR UART10EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR ADC1EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR ADC2EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR ADC3EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR SDIOEN LL_APB2_GRP1_IsEnabledClock
- APB2ENR SPI1EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR SPI4EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR SYSCFGEN LL_APB2_GRP1_IsEnabledClock
- APB2ENR EXTITEN LL_APB2_GRP1_IsEnabledClock
- APB2ENR TIM9EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR TIM10EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR TIM11EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR SPI5EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR SPI6EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR SAI1EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR SAI2EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR LTDCEN LL_APB2_GRP1_IsEnabledClock
- APB2ENR DSIEN LL_APB2_GRP1_IsEnabledClock
- APB2ENR DFSDM1EN LL_APB2_GRP1_IsEnabledClock
- APB2ENR DFSDM2EN LL_APB2_GRP1_IsEnabledClock

LL_APB2_GRP1_DisableClock

Function name

__STATIC_INLINE void LL_APB2_GRP1_DisableClock (uint32_t Periphs)

Function description

Disable APB2 peripherals clock.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB2_GRP1_PERIPH_TIM1
  - LL_APB2_GRP1_PERIPH_TIM8 (*)
  - LL_APB2_GRP1_PERIPH_USART1
  - LL_APB2_GRP1_PERIPH_USART6 (*)
  - LL_APB2_GRP1_PERIPH_UART9 (*)
  - LL_APB2_GRP1_PERIPH_UART10 (*)
  - LL_APB2_GRP1_PERIPH_ADC1
  - LL_APB2_GRP1_PERIPH_ADC2 (*)
  - LL_APB2_GRP1_PERIPH_ADC3 (*)
  - LL_APB2_GRP1_PERIPH_SDIO (*)
  - LL_APB2_GRP1_PERIPH_SPI1
  - LL_APB2_GRP1_PERIPH_SPI4 (*)
  - LL_APB2_GRP1_PERIPH_SYSCFG
  - LL_APB2_GRP1_PERIPH_EXTI (*)
  - LL_APB2_GRP1_PERIPH_TIM9
  - LL_APB2_GRP1_PERIPH_TIM10 (*)
  - LL_APB2_GRP1_PERIPH_TIM11
  - LL_APB2_GRP1_PERIPH_SPI5 (*)
  - LL_APB2_GRP1_PERIPH_SPI6 (*)
  - LL_APB2_GRP1_PERIPH_SAI1 (*)
  - LL_APB2_GRP1_PERIPH_SAI2 (*)
  - LL_APB2_GRP1_PERIPH_LTDC (*)
  - LL_APB2_GRP1_PERIPH_DSI (*)
  - LL_APB2_GRP1_PERIPH_DFSDM1 (*)
  - LL_APB2_GRP1_PERIPH_DFSDM2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB2ENR TIM1EN LL_APB2_GRP1_DisableClock
- APB2ENR TIM8EN LL_APB2_GRP1_DisableClock
- APB2ENR USART1EN LL_APB2_GRP1_DisableClock
- APB2ENR USART6EN LL_APB2_GRP1_DisableClock
- APB2ENR UART9EN LL_APB2_GRP1_DisableClock
- APB2ENR UART10EN LL_APB2_GRP1_DisableClock
- APB2ENR ADC1EN LL_APB2_GRP1_DisableClock
- APB2ENR ADC2EN LL_APB2_GRP1_DisableClock
- APB2ENR ADC3EN LL_APB2_GRP1_DisableClock
- APB2ENR SDIOEN LL_APB2_GRP1_DisableClock
- APB2ENR SPI1EN LL_APB2_GRP1_DisableClock
- APB2ENR SPI4EN LL_APB2_GRP1_DisableClock
- APB2ENR SYSCFGEN LL_APB2_GRP1_DisableClock
- APB2ENR EXTITEN LL_APB2_GRP1_DisableClock
- APB2ENR TIM9EN LL_APB2_GRP1_DisableClock
- APB2ENR TIM10EN LL_APB2_GRP1_DisableClock
- APB2ENR TIM11EN LL_APB2_GRP1_DisableClock
- APB2ENR SPI5EN LL_APB2_GRP1_DisableClock
- APB2ENR SPI6EN LL_APB2_GRP1_DisableClock
- APB2ENR SAI1EN LL_APB2_GRP1_DisableClock
- APB2ENR SAI2EN LL_APB2_GRP1_DisableClock
- APB2ENR LTDCEN LL_APB2_GRP1_DisableClock
- APB2ENR DSIEN LL_APB2_GRP1_DisableClock
- APB2ENR DFSDM1EN LL_APB2_GRP1_DisableClock
- APB2ENR DFSDM2EN LL_APB2_GRP1_DisableClock

LL_APB2_GRP1_ForceReset

Function name

__STATIC_INLINE void LL_APB2_GRP1_ForceReset (uint32_t Periphs)

Function description

Force APB2 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB2_GRP1_PERIPH_ALL
  - LL_APB2_GRP1_PERIPH_TIM1
  - LL_APB2_GRP1_PERIPH_TIM8 (*)
  - LL_APB2_GRP1_PERIPH_USART1
  - LL_APB2_GRP1_PERIPH_USART6 (*)
  - LL_APB2_GRP1_PERIPH_UART9 (*)
  - LL_APB2_GRP1_PERIPH_UART10 (*)
  - LL_APB2_GRP1_PERIPH_ADC
  - LL_APB2_GRP1_PERIPH_SDIO (*)
  - LL_APB2_GRP1_PERIPH_SPI1
  - LL_APB2_GRP1_PERIPH_SPI4 (*)
  - LL_APB2_GRP1_PERIPH_SYSCFG
  - LL_APB2_GRP1_PERIPH_TIM9
  - LL_APB2_GRP1_PERIPH_TIM10 (*)
  - LL_APB2_GRP1_PERIPH_TIM11
  - LL_APB2_GRP1_PERIPH_SPI5 (*)
  - LL_APB2_GRP1_PERIPH_SPI6 (*)
  - LL_APB2_GRP1_PERIPH_SAI1 (*)
  - LL_APB2_GRP1_PERIPH_SAI2 (*)
  - LL_APB2_GRP1_PERIPH_LTDC (*)
  - LL_APB2_GRP1_PERIPH_DSI (*)
  - LL_APB2_GRP1_PERIPH_DFSDM1 (*)
  - LL_APB2_GRP1_PERIPH_DFSDM2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB2RSTR TIM1RST LL_APB2_GRP1_ForceReset
- APB2RSTR TIM8RST LL_APB2_GRP1_ForceReset
- APB2RSTR USART1RST LL_APB2_GRP1_ForceReset
- APB2RSTR USART6RST LL_APB2_GRP1_ForceReset
- APB2RSTR UART9RST LL_APB2_GRP1_ForceReset
- APB2RSTR UART10RST LL_APB2_GRP1_ForceReset
- APB2RSTR ADCRST LL_APB2_GRP1_ForceReset
- APB2RSTR SDIORST LL_APB2_GRP1_ForceReset
- APB2RSTR SPI1RST LL_APB2_GRP1_ForceReset
- APB2RSTR SPI4RST LL_APB2_GRP1_ForceReset
- APB2RSTR SYSCFGRST LL_APB2_GRP1_ForceReset
- APB2RSTR TIM9RST LL_APB2_GRP1_ForceReset
- APB2RSTR TIM10RST LL_APB2_GRP1_ForceReset
- APB2RSTR TIM11RST LL_APB2_GRP1_ForceReset
- APB2RSTR SPI5RST LL_APB2_GRP1_ForceReset
- APB2RSTR SPI6RST LL_APB2_GRP1_ForceReset
- APB2RSTR SAI1RST LL_APB2_GRP1_ForceReset
- APB2RSTR SAI2RST LL_APB2_GRP1_ForceReset
- APB2RSTR LTDCRST LL_APB2_GRP1_ForceReset
- APB2RSTR DSIRST LL_APB2_GRP1_ForceReset
- APB2RSTR DFSDM1RST LL_APB2_GRP1_ForceReset
- APB2RSTR DFSDM2RST LL_APB2_GRP1_ForceReset

LL_APB2_GRP1_ReleaseReset

Function name

__STATIC_INLINE void LL_APB2_GRP1_ReleaseReset (uint32_t Periphs)

Function description

Release APB2 peripherals reset.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB2_GRP1_PERIPH_ALL
  - LL_APB2_GRP1_PERIPH_TIM1
  - LL_APB2_GRP1_PERIPH_TIM8 (*)
  - LL_APB2_GRP1_PERIPH_USART1
  - LL_APB2_GRP1_PERIPH_USART6 (*)
  - LL_APB2_GRP1_PERIPH_UART9 (*)
  - LL_APB2_GRP1_PERIPH_UART10 (*)
  - LL_APB2_GRP1_PERIPH_ADC
  - LL_APB2_GRP1_PERIPH_SDIO (*)
  - LL_APB2_GRP1_PERIPH_SPI1
  - LL_APB2_GRP1_PERIPH_SPI4 (*)
  - LL_APB2_GRP1_PERIPH_SYSCFG
  - LL_APB2_GRP1_PERIPH_EXTI (*)
  - LL_APB2_GRP1_PERIPH_TIM9
  - LL_APB2_GRP1_PERIPH_TIM10 (*)
  - LL_APB2_GRP1_PERIPH_TIM11
  - LL_APB2_GRP1_PERIPH_SPI5 (*)
  - LL_APB2_GRP1_PERIPH_SPI6 (*)
  - LL_APB2_GRP1_PERIPH_SAI1 (*)
  - LL_APB2_GRP1_PERIPH_SAI2 (*)
  - LL_APB2_GRP1_PERIPH_LTDC (*)
  - LL_APB2_GRP1_PERIPH_DSI (*)
  - LL_APB2_GRP1_PERIPH_DFSDM1 (*)
  - LL_APB2_GRP1_PERIPH_DFSDM2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB2RSTR TIM1RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR TIM8RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR USART1RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR USART6RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR UART9RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR UART10RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR ADCRST LL_APB2_GRP1_ReleaseReset
- APB2RSTR SDIORST LL_APB2_GRP1_ReleaseReset
- APB2RSTR SPI1RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR SPI4RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR SYSCFGRST LL_APB2_GRP1_ReleaseReset
- APB2RSTR TIM9RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR TIM10RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR TIM11RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR SPI5RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR SPI6RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR SAI1RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR SAI2RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR LTDCRST LL_APB2_GRP1_ReleaseReset
- APB2RSTR DSIRST LL_APB2_GRP1_ReleaseReset
- APB2RSTR DFSDM1RST LL_APB2_GRP1_ReleaseReset
- APB2RSTR DFSDM2RST LL_APB2_GRP1_ReleaseReset

LL_APB2_GRP1_EnableClockLowPower

Function name

__STATIC_INLINE void LL_APB2_GRP1_EnableClockLowPower (uint32_t Periphs)

Function description

Enable APB2 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB2_GRP1_PERIPH_TIM1
  - LL_APB2_GRP1_PERIPH_TIM8 (*)
  - LL_APB2_GRP1_PERIPH_USART1
  - LL_APB2_GRP1_PERIPH_USART6 (*)
  - LL_APB2_GRP1_PERIPH_UART9 (*)
  - LL_APB2_GRP1_PERIPH_UART10 (*)
  - LL_APB2_GRP1_PERIPH_ADC1
  - LL_APB2_GRP1_PERIPH_ADC2 (*)
  - LL_APB2_GRP1_PERIPH_ADC3 (*)
  - LL_APB2_GRP1_PERIPH_SDIO (*)
  - LL_APB2_GRP1_PERIPH_SPI1
  - LL_APB2_GRP1_PERIPH_SPI4 (*)
  - LL_APB2_GRP1_PERIPH_SYSCFG
  - LL_APB2_GRP1_PERIPH_EXTI (*)
  - LL_APB2_GRP1_PERIPH_TIM9
  - LL_APB2_GRP1_PERIPH_TIM10 (*)
  - LL_APB2_GRP1_PERIPH_TIM11
  - LL_APB2_GRP1_PERIPH_SPI5 (*)
  - LL_APB2_GRP1_PERIPH_SPI6 (*)
  - LL_APB2_GRP1_PERIPH_SAI1 (*)
  - LL_APB2_GRP1_PERIPH_SAI2 (*)
  - LL_APB2_GRP1_PERIPH_LTDC (*)
  - LL_APB2_GRP1_PERIPH_DSI (*)
  - LL_APB2_GRP1_PERIPH_DFSDM1 (*)
  - LL_APB2_GRP1_PERIPH_DFSDM2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB2LPENR TIM1LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR TIM8LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR USART1LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR USART6LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR UART9LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR UART10LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR ADC1LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR ADC2LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR ADC3LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR SDIOLPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR SPI1LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR SPI4LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR SYSCFGLPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR EXTITLPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR TIM9LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR TIM10LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR TIM11LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR SPI5LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR SPI6LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR SAI1LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR SAI2LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR LTDCLPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR DSILPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR DFSDM1LPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR DSILPEN LL_APB2_GRP1_EnableClockLowPower
- APB2LPENR DFSDM2LPEN LL_APB2_GRP1_EnableClockLowPower

LL_APB2_GRP1_DisableClockLowPower

Function name

__STATIC_INLINE void LL_APB2_GRP1_DisableClockLowPower (uint32_t Periphs)

Function description

Disable APB2 peripheral clocks in low-power mode.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_APB2_GRP1_PERIPH_TIM1
  - LL_APB2_GRP1_PERIPH_TIM8 (*)
  - LL_APB2_GRP1_PERIPH_USART1
  - LL_APB2_GRP1_PERIPH_USART6 (*)
  - LL_APB2_GRP1_PERIPH_UART9 (*)
  - LL_APB2_GRP1_PERIPH_UART10 (*)
  - LL_APB2_GRP1_PERIPH_ADC1
  - LL_APB2_GRP1_PERIPH_ADC2 (*)
  - LL_APB2_GRP1_PERIPH_ADC3 (*)
  - LL_APB2_GRP1_PERIPH_SDIO (*)
  - LL_APB2_GRP1_PERIPH_SPI1
  - LL_APB2_GRP1_PERIPH_SPI4 (*)
  - LL_APB2_GRP1_PERIPH_SYSCFG
  - LL_APB2_GRP1_PERIPH_EXTI (*)
  - LL_APB2_GRP1_PERIPH_TIM9
  - LL_APB2_GRP1_PERIPH_TIM10 (*)
  - LL_APB2_GRP1_PERIPH_TIM11
  - LL_APB2_GRP1_PERIPH_SPI5 (*)
  - LL_APB2_GRP1_PERIPH_SPI6 (*)
  - LL_APB2_GRP1_PERIPH_SAI1 (*)
  - LL_APB2_GRP1_PERIPH_SAI2 (*)
  - LL_APB2_GRP1_PERIPH_LTDC (*)
  - LL_APB2_GRP1_PERIPH_DSI (*)
  - LL_APB2_GRP1_PERIPH_DFSDM1 (*)
  - LL_APB2_GRP1_PERIPH_DFSDM2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- APB2LPENR TIM1LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR TIM8LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR USART1LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR USART6LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR UART9LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR UART10LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR ADC1LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR ADC2LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR ADC3LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR SDIOLPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR SPI1LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR SPI4LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR SYSCFGLPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR EXTITLPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR TIM9LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR TIM10LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR TIM11LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR SPI5LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR SPI6LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR SAI1LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR SAI2LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR LTDCLPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR DSILPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR DFSDM1LPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR DSILPEN LL_APB2_GRP1_DisableClockLowPower
- APB2LPENR DFSDM2LPEN LL_APB2_GRP1_DisableClockLowPower

74.2 BUS Firmware driver defines

The following section lists the various define and macros of the module.

74.2.1 BUS

BUS

AHB1 GRP1 PERIPH

LL_AHB1_GRP1_PERIPH_ALL

LL_AHB1_GRP1_PERIPH_GPIOA

LL_AHB1_GRP1_PERIPH_GPIOB

LL_AHB1_GRP1_PERIPH_GPIOC

LL_AHB1_GRP1_PERIPH_GPIOD

LL_AHB1_GRP1_PERIPH_GPIOE

LL_AHB1_GRP1_PERIPH_GPIOF

LL_AHB1_GRP1_PERIPH_GPIOG

LL_AHB1_GRP1_PERIPH_GPIOH

LL_AHB1_GRP1_PERIPH_GPIOI

LL_AHB1_GRP1_PERIPH_GPIOJ

LL_AHB1_GRP1_PERIPH_GPIOK

LL_AHB1_GRP1_PERIPH_CRC

LL_AHB1_GRP1_PERIPH_BKPSRAM

LL_AHB1_GRP1_PERIPH_CCMDATARAM

LL_AHB1_GRP1_PERIPH_DMA1

LL_AHB1_GRP1_PERIPH_DMA2

LL_AHB1_GRP1_PERIPH_DMA2D

LL_AHB1_GRP1_PERIPH_ETHMAC

LL_AHB1_GRP1_PERIPH_ETHMACTX

LL_AHB1_GRP1_PERIPH_ETHMACRX

LL_AHB1_GRP1_PERIPH_ETHMACPTP

LL_AHB1_GRP1_PERIPH_OTGHS

LL_AHB1_GRP1_PERIPH_OTGHSULPI

LL_AHB1_GRP1_PERIPH_FLITF

LL_AHB1_GRP1_PERIPH_SRAM1

LL_AHB1_GRP1_PERIPH_SRAM2

LL_AHB1_GRP1_PERIPH_SRAM3

AHB2 GRP1 PERIPH

LL_AHB2_GRP1_PERIPH_ALL

LL_AHB2_GRP1_PERIPH_DCMI

LL_AHB2_GRP1_PERIPH_CRYP

LL_AHB2_GRP1_PERIPH_HASH

LL_AHB2_GRP1_PERIPH_RNG

LL_AHB2_GRP1_PERIPH_OTGFS

AHB3 GRP1 PERIPH

LL_AHB3_GRP1_PERIPH_ALL

LL_AHB3_GRP1_PERIPH_FMC

LL_AHB3_GRP1_PERIPH_QSPI

APB1 GRP1 PERIPH

LL_APB1_GRP1_PERIPH_ALL

LL_APB1_GRP1_PERIPH_TIM2

LL_APB1_GRP1_PERIPH_TIM3

LL_APB1_GRP1_PERIPH_TIM4

LL_APB1_GRP1_PERIPH_TIM5

LL_APB1_GRP1_PERIPH_TIM6

LL_APB1_GRP1_PERIPH_TIM7

LL_APB1_GRP1_PERIPH_TIM12

LL_APB1_GRP1_PERIPH_TIM13

LL_APB1_GRP1_PERIPH_TIM14

LL_APB1_GRP1_PERIPH_WWDG

LL_APB1_GRP1_PERIPH_SPI2

LL_APB1_GRP1_PERIPH_SPI3

LL_APB1_GRP1_PERIPH_USART2

LL_APB1_GRP1_PERIPH_USART3

LL_APB1_GRP1_PERIPH_UART4

LL_APB1_GRP1_PERIPH_UART5

LL_APB1_GRP1_PERIPH_I2C1

LL_APB1_GRP1_PERIPH_I2C2

LL_APB1_GRP1_PERIPH_I2C3

LL_APB1_GRP1_PERIPH_CAN1

LL_APB1_GRP1_PERIPH_CAN2

LL_APB1_GRP1_PERIPH_PWR

LL_APB1_GRP1_PERIPH_DAC1

LL_APB1_GRP1_PERIPH_UART7

LL_APB1_GRP1_PERIPH_UART8

APB2 GRP1 PERIPH

LL_APB2_GRP1_PERIPH_ALL

LL_APB2_GRP1_PERIPH_TIM1

LL_APB2_GRP1_PERIPH_TIM8

LL_APB2_GRP1_PERIPH_USART1

LL_APB2_GRP1_PERIPH_USART6

LL_APB2_GRP1_PERIPH_ADC1

LL_APB2_GRP1_PERIPH_ADC2

LL_APB2_GRP1_PERIPH_ADC3

LL_APB2_GRP1_PERIPH_SDIO

LL_APB2_GRP1_PERIPH_SPI1

LL_APB2_GRP1_PERIPH_SPI4

LL_APB2_GRP1_PERIPH_SYSCFG

LL_APB2_GRP1_PERIPH_TIM9

LL_APB2_GRP1_PERIPH_TIM10

LL_APB2_GRP1_PERIPH_TIM11

LL_APB2_GRP1_PERIPH_SPI5

LL_APB2_GRP1_PERIPH_SPI6

LL_APB2_GRP1_PERIPH_SAI1

LL_APB2_GRP1_PERIPH_LTDC

LL_APB2_GRP1_PERIPH_DSI

LL_APB2_GRP1_PERIPH_ADC

75 LL CORTEX Generic Driver

75.1 CORTEX Firmware driver API description

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

75.1.1 Detailed description of functions

LL_SYSTICK_IsActiveCounterFlag

Function name

__STATIC_INLINE uint32_t LL_SYSTICK_IsActiveCounterFlag (void)

Function description

This function checks if the Systick counter flag is active or not.

Return values

- **State:** of bit (1 or 0).

Notes

- It can be used in timeout function on application side.

Reference Manual to LL API cross reference:

- STK_CTRL COUNTFLAG LL_SYSTICK_IsActiveCounterFlag

LL_SYSTICK_SetClkSource

Function name

__STATIC_INLINE void LL_SYSTICK_SetClkSource (uint32_t Source)

Function description

Configures the SysTick clock source.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_SYSTICK_CLKSOURCE_HCLK_DIV8
  - LL_SYSTICK_CLKSOURCE_HCLK

Return values

- **None:**

Reference Manual to LL API cross reference:

- STK_CTRL CLKSOURCE LL_SYSTICK_SetClkSource

LL_SYSTICK_GetClkSource

Function name

__STATIC_INLINE uint32_t LL_SYSTICK_GetClkSource (void)

Function description

Get the SysTick clock source.

Return values

- **Returned:** value can be one of the following values:
  - LL_SYSTICK_CLKSOURCE_HCLK_DIV8
  - LL_SYSTICK_CLKSOURCE_HCLK

Reference Manual to LL API cross reference:

- STK_CTRL CLKSOURCE LL_SYSTICK_GetClkSource

LL_SYSTICK_EnableIT

Function name

__STATIC_INLINE void LL_SYSTICK_EnableIT (void)

Function description

Enable SysTick exception request.

Return values

- **None:**

Reference Manual to LL API cross reference:

- STK_CTRL TICKINT LL_SYSTICK_EnableIT

LL_SYSTICK_DisableIT

Function name

__STATIC_INLINE void LL_SYSTICK_DisableIT (void)

Function description

Disable SysTick exception request.

Return values

- **None:**

Reference Manual to LL API cross reference:

- STK_CTRL TICKINT LL_SYSTICK_DisableIT

LL_SYSTICK_IsEnabledIT

Function name

__STATIC_INLINE uint32_t LL_SYSTICK_IsEnabledIT (void)

Function description

Checks if the SYSTICK interrupt is enabled or disabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- STK_CTRL TICKINT LL_SYSTICK_IsEnabledIT

LL_LPM_EnableSleep

Function name

__STATIC_INLINE void LL_LPM_EnableSleep (void)

Function description

Processor uses sleep as its low power mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SCB_SCR SLEEPDEEP LL_LPM_EnableSleep

LL_LPM_EnableDeepSleep

Function name

__STATIC_INLINE void LL_LPM_EnableDeepSleep (void)

Function description

Processor uses deep sleep as its low power mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SCB_SCR SLEEPDEEP LL_LPM_EnableDeepSleep

LL_LPM_EnableSleepOnExit

Function name

__STATIC_INLINE void LL_LPM_EnableSleepOnExit (void)

Function description

Configures sleep-on-exit when returning from Handler mode to Thread mode.

Return values

- **None:**

Notes

- Setting this bit to 1 enables an interrupt-driven application to avoid returning to an empty main application.

Reference Manual to LL API cross reference:

- SCB_SCR SLEEPONEXIT LL_LPM_EnableSleepOnExit

LL_LPM_DisableSleepOnExit

Function name

__STATIC_INLINE void LL_LPM_DisableSleepOnExit (void)

Function description

Do not sleep when returning to Thread mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SCB_SCR SLEEPONEXIT LL_LPM_DisableSleepOnExit

LL_LPM_EnableEventOnPend

Function name

__STATIC_INLINE void LL_LPM_EnableEventOnPend (void)

Function description

Enabled events and all interrupts, including disabled interrupts, can wakeup the processor.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SCB_SCR SEVEONPEND LL_LPM_EnableEventOnPend

LL_LPM_DisableEventOnPend

Function name

__STATIC_INLINE void LL_LPM_DisableEventOnPend (void)

Function description

Only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SCB_SCR SEVEONPEND LL_LPM_DisableEventOnPend

LL_HANDLER_EnableFault

Function name

__STATIC_INLINE void LL_HANDLER_EnableFault (uint32_t Fault)

Function description

Enable a fault in System handler control register (SHCSR)

Parameters

- **Fault:** This parameter can be a combination of the following values:
  - LL_HANDLER_FAULT_USG
  - LL_HANDLER_FAULT_BUS
  - LL_HANDLER_FAULT_MEM

Return values

- **None:**

Reference Manual to LL API cross reference:

- SCB_SHCSR MEMFAULTENA LL_HANDLER_EnableFault

LL_HANDLER_DisableFault

Function name

__STATIC_INLINE void LL_HANDLER_DisableFault (uint32_t Fault)

Function description

Disable a fault in System handler control register (SHCSR)

Parameters

- **Fault:** This parameter can be a combination of the following values:
  - LL_HANDLER_FAULT_USG
  - LL_HANDLER_FAULT_BUS
  - LL_HANDLER_FAULT_MEM

Return values

- **None:**

Reference Manual to LL API cross reference:

- SCB_SHCSR MEMFAULTENA LL_HANDLER_DisableFault

LL_CPUID_GetImplementer

Function name

__STATIC_INLINE uint32_t LL_CPUID_GetImplementer (void)

Function description

Get Implementer code.

Return values

- **Value:** should be equal to 0x41 for ARM

Reference Manual to LL API cross reference:

- SCB_CPUID IMPLEMENTER LL_CPUID_GetImplementer

LL_CPUID_GetVariant

Function name

__STATIC_INLINE uint32_t LL_CPUID_GetVariant (void)

Function description

Get Variant number (The r value in the rnpn product revision identifier)

Return values

- **Value:** between 0 and 255 (0x0: revision 0)

Reference Manual to LL API cross reference:

- SCB_CPUID VARIANT LL_CPUID_GetVariant

LL_CPUID_GetConstant

Function name

__STATIC_INLINE uint32_t LL_CPUID_GetConstant (void)

Function description

Get Constant number.

Return values

- **Value:** should be equal to 0xF for Cortex-M4 devices

Reference Manual to LL API cross reference:

- SCB_CPUID ARCHITECTURE LL_CPUID_GetConstant

LL_CPUID_GetParNo

Function name

__STATIC_INLINE uint32_t LL_CPUID_GetParNo (void)

Function description

Get Part number.

Return values

- **Value:** should be equal to 0xC24 for Cortex-M4

Reference Manual to LL API cross reference:

- SCB_CPUID PARTNO LL_CPUID_GetParNo

LL_CPUID_GetRevision

Function name

__STATIC_INLINE uint32_t LL_CPUID_GetRevision (void)

Function description

Get Revision number (The p value in the rnpn product revision identifier, indicates patch release)

Return values

- **Value:** between 0 and 255 (0x1: patch 1)

Reference Manual to LL API cross reference:

- SCB_CPUID REVISION LL_CPUID_GetRevision

LL_MPU_Enable

Function name

__STATIC_INLINE void LL_MPU_Enable (uint32_t Options)

Function description

Enable MPU with input options.

Parameters

- **Options:** This parameter can be one of the following values:
  - LL_MPU_CTRL_HFNMI_PRIVDEF_NONE
  - LL_MPU_CTRL_HARDFAULT_NMI
  - LL_MPU_CTRL_PRIVILEGED_DEFAULT
  - LL_MPU_CTRL_HFNMI_PRIVDEF

Return values

- **None:**

Reference Manual to LL API cross reference:

- MPU_CTRL ENABLE LL_MPU_Enable

LL_MPU_Disable

Function name

__STATIC_INLINE void LL_MPU_Disable (void)

Function description

Disable MPU.

Return values

- **None:**

Reference Manual to LL API cross reference:

- MPU_CTRL ENABLE LL_MPU_Disable

LL_MPU_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_MPU_IsEnabled (void)

Function description

Check if MPU is enabled or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- MPU_CTRL ENABLE LL_MPU_IsEnabled

LL_MPU_EnableRegion

Function name

__STATIC_INLINE void LL_MPU_EnableRegion (uint32_t Region)

Function description

Enable a MPU region.

Parameters

- **Region:** This parameter can be one of the following values:
  - LL_MPU_REGION_NUMBER0
  - LL_MPU_REGION_NUMBER1
  - LL_MPU_REGION_NUMBER2
  - LL_MPU_REGION_NUMBER3
  - LL_MPU_REGION_NUMBER4
  - LL_MPU_REGION_NUMBER5
  - LL_MPU_REGION_NUMBER6
  - LL_MPU_REGION_NUMBER7

Return values

- **None:**

Reference Manual to LL API cross reference:

- MPU_RASR ENABLE LL_MPU_EnableRegion

LL_MPU_ConfigRegion

Function name

__STATIC_INLINE void LL_MPU_ConfigRegion (uint32_t Region, uint32_t SubRegionDisable, uint32_t Address, uint32_t Attributes)

Function description

Configure and enable a region.

Parameters

- **Region:** This parameter can be one of the following values:
  - LL_MPU_REGION_NUMBER0
  - LL_MPU_REGION_NUMBER1
  - LL_MPU_REGION_NUMBER2
  - LL_MPU_REGION_NUMBER3
  - LL_MPU_REGION_NUMBER4
  - LL_MPU_REGION_NUMBER5
  - LL_MPU_REGION_NUMBER6
  - LL_MPU_REGION_NUMBER7
- **Address:** Value of region base address
- **SubRegionDisable:** Sub-region disable value between Min_Data = 0x00 and Max_Data = 0xFF
- **Attributes:** This parameter can be a combination of the following values:
  - LL_MPU_REGION_SIZE_32B or LL_MPU_REGION_SIZE_64B or LL_MPU_REGION_SIZE_128B or LL_MPU_REGION_SIZE_256B or LL_MPU_REGION_SIZE_512B or LL_MPU_REGION_SIZE_1KB or LL_MPU_REGION_SIZE_2KB or LL_MPU_REGION_SIZE_4KB or LL_MPU_REGION_SIZE_8KB or LL_MPU_REGION_SIZE_16KB or LL_MPU_REGION_SIZE_32KB or LL_MPU_REGION_SIZE_64KB or LL_MPU_REGION_SIZE_128KB or LL_MPU_REGION_SIZE_256KB or LL_MPU_REGION_SIZE_512KB or LL_MPU_REGION_SIZE_1MB or LL_MPU_REGION_SIZE_2MB or LL_MPU_REGION_SIZE_4MB or LL_MPU_REGION_SIZE_8MB or LL_MPU_REGION_SIZE_16MB or LL_MPU_REGION_SIZE_32MB or LL_MPU_REGION_SIZE_64MB or LL_MPU_REGION_SIZE_128MB or LL_MPU_REGION_SIZE_256MB or LL_MPU_REGION_SIZE_512MB or LL_MPU_REGION_SIZE_1GB or LL_MPU_REGION_SIZE_2GB or LL_MPU_REGION_SIZE_4GB
  - LL_MPU_REGION_NO_ACCESS or LL_MPU_REGION_PRIV_RW or LL_MPU_REGION_PRIV_RW_URO or LL_MPU_REGION_FULL_ACCESS or LL_MPU_REGION_PRIV_RO or LL_MPU_REGION_PRIV_RO_URO
  - LL_MPU_TEX_LEVEL0 or LL_MPU_TEX_LEVEL1 or LL_MPU_TEX_LEVEL2 or LL_MPU_TEX_LEVEL4
  - LL_MPU_INSTRUCTION_ACCESS_ENABLE or LL_MPU_INSTRUCTION_ACCESS_DISABLE
  - LL_MPU_ACCESS_SHAREABLE or LL_MPU_ACCESS_NOT_SHAREABLE
  - LL_MPU_ACCESS_CACHEABLE or LL_MPU_ACCESS_NOT_CACHEABLE
  - LL_MPU_ACCESS_BUFFERABLE or LL_MPU_ACCESS_NOT_BUFFERABLE

Return values

- **None:**

Reference Manual to LL API cross reference:

- MPU_RNR REGION LL_MPU_ConfigRegion
- MPU_RBAR REGION LL_MPU_ConfigRegion
- MPU_RBAR ADDR LL_MPU_ConfigRegion
- MPU_RASR XN LL_MPU_ConfigRegion
- MPU_RASR AP LL_MPU_ConfigRegion
- MPU_RASR S LL_MPU_ConfigRegion
- MPU_RASR C LL_MPU_ConfigRegion
- MPU_RASR B LL_MPU_ConfigRegion
- MPU_RASR SIZE LL_MPU_ConfigRegion

LL_MPU_DisableRegion

Function name

__STATIC_INLINE void LL_MPU_DisableRegion (uint32_t Region)

Function description

Disable a region.

Parameters

- **Region:** This parameter can be one of the following values:
  - LL_MPU_REGION_NUMBER0
  - LL_MPU_REGION_NUMBER1
  - LL_MPU_REGION_NUMBER2
  - LL_MPU_REGION_NUMBER3
  - LL_MPU_REGION_NUMBER4
  - LL_MPU_REGION_NUMBER5
  - LL_MPU_REGION_NUMBER6
  - LL_MPU_REGION_NUMBER7

Return values

- **None:**

Reference Manual to LL API cross reference:

- MPU_RNR REGION LL_MPU_DisableRegion
- MPU_RASR ENABLE LL_MPU_DisableRegion

75.2 CORTEX Firmware driver defines

The following section lists the various define and macros of the module.

75.2.1 CORTEX

CORTEX

MPU Bufferable Access

LL_MPU_ACCESS_BUFFERABLE

Bufferable memory attribute

LL_MPU_ACCESS_NOT_BUFFERABLE

Not Bufferable memory attribute

MPU Cacheable Access

LL_MPU_ACCESS_CACHEABLE

Cacheable memory attribute

LL_MPU_ACCESS_NOT_CACHEABLE

Not Cacheable memory attribute

SYSTICK Clock Source

LL_SYSTICK_CLKSOURCE_HCLK_DIV8

AHB clock divided by 8 selected as SysTick clock source.

LL_SYSTICK_CLKSOURCE_HCLK

AHB clock selected as SysTick clock source.

MPU Control

LL_MPU_CTRL_HFNMI_PRIVDEF_NONE

Disable NMI and privileged SW access

LL_MPU_CTRL_HARDFAULT_NMI

Enables the operation of MPU during hard fault, NMI, and FAULTMASK handlers

LL_MPU_CTRL_PRIVILEGED_DEFAULT

Enable privileged software access to default memory map

LL_MPU_CTRL_HFNMI_PRIVDEF

Enable NMI and privileged SW access

Handler Fault type

LL_HANDLER_FAULT_USG

Usage fault

LL_HANDLER_FAULT_BUS

Bus fault

LL_HANDLER_FAULT_MEM

Memory management fault

MPU Instruction Access

LL_MPU_INSTRUCTION_ACCESS_ENABLE

Instruction fetches enabled

LL_MPU_INSTRUCTION_ACCESS_DISABLE

Instruction fetches disabled

MPU Region Number

LL_MPU_REGION_NUMBER0

REGION Number 0

LL_MPU_REGION_NUMBER1

REGION Number 1

LL_MPU_REGION_NUMBER2

REGION Number 2

LL_MPU_REGION_NUMBER3

REGION Number 3

LL_MPU_REGION_NUMBER4

REGION Number 4

LL_MPU_REGION_NUMBER5

REGION Number 5

LL_MPU_REGION_NUMBER6

REGION Number 6

LL_MPU_REGION_NUMBER7

REGION Number 7

MPU Region Privileges

LL_MPU_REGION_NO_ACCESS

No access

LL_MPU_REGION_PRIV_RW

RW privileged (privileged access only)

LL_MPU_REGION_PRIV_RW_URO

RW privileged - RO user (Write in a user program generates a fault)

LL_MPU_REGION_FULL_ACCESS

RW privileged & user (Full access)

LL_MPU_REGION_PRIV_RO

RO privileged (privileged read only)

LL_MPU_REGION_PRIV_RO_URO

RO privileged & user (read only)

MPU Region Size

LL_MPU_REGION_SIZE_32B

32B Size of the MPU protection region

LL_MPU_REGION_SIZE_64B

64B Size of the MPU protection region

LL_MPU_REGION_SIZE_128B

128B Size of the MPU protection region

LL_MPU_REGION_SIZE_256B

256B Size of the MPU protection region

LL_MPU_REGION_SIZE_512B

512B Size of the MPU protection region

LL_MPU_REGION_SIZE_1KB

1KB Size of the MPU protection region

LL_MPU_REGION_SIZE_2KB

2KB Size of the MPU protection region

LL_MPU_REGION_SIZE_4KB

4KB Size of the MPU protection region

LL_MPU_REGION_SIZE_8KB

8KB Size of the MPU protection region

LL_MPU_REGION_SIZE_16KB

16KB Size of the MPU protection region

LL_MPU_REGION_SIZE_32KB

32KB Size of the MPU protection region

LL_MPU_REGION_SIZE_64KB

64KB Size of the MPU protection region

LL_MPU_REGION_SIZE_128KB

128KB Size of the MPU protection region

LL_MPU_REGION_SIZE_256KB

256KB Size of the MPU protection region

LL_MPU_REGION_SIZE_512KB

512KB Size of the MPU protection region

LL_MPU_REGION_SIZE_1MB

1MB Size of the MPU protection region

LL_MPU_REGION_SIZE_2MB

2MB Size of the MPU protection region

LL_MPU_REGION_SIZE_4MB

4MB Size of the MPU protection region

LL_MPU_REGION_SIZE_8MB

8MB Size of the MPU protection region

LL_MPU_REGION_SIZE_16MB

16MB Size of the MPU protection region

LL_MPU_REGION_SIZE_32MB

32MB Size of the MPU protection region

LL_MPU_REGION_SIZE_64MB

64MB Size of the MPU protection region

LL_MPU_REGION_SIZE_128MB

128MB Size of the MPU protection region

LL_MPU_REGION_SIZE_256MB

256MB Size of the MPU protection region

LL_MPU_REGION_SIZE_512MB

512MB Size of the MPU protection region

LL_MPU_REGION_SIZE_1GB

1GB Size of the MPU protection region

LL_MPU_REGION_SIZE_2GB

2GB Size of the MPU protection region

LL_MPU_REGION_SIZE_4GB

4GB Size of the MPU protection region

MPU Shareable Access

LL_MPU_ACCESS_SHAREABLE

Shareable memory attribute

LL_MPU_ACCESS_NOT_SHAREABLE

Not Shareable memory attribute

MPU TEX Level

LL_MPU_TEX_LEVEL0

b000 for TEX bits

LL_MPU_TEX_LEVEL1

b001 for TEX bits

LL_MPU_TEX_LEVEL2

b010 for TEX bits

LL_MPU_TEX_LEVEL4

b100 for TEX bits

76 LL CRC Generic Driver

76.1 CRC Firmware driver API description

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

76.1.1 Detailed description of functions

LL_CRC_ResetCRCCalculationUnit

Function name

__STATIC_INLINE void LL_CRC_ResetCRCCalculationUnit (CRC_TypeDef * CRCx)

Function description

Reset the CRC calculation unit.

Parameters

- **CRCx:** CRC Instance

Return values

- **None:**

Notes

- If Programmable Initial CRC value feature is available, also set the Data Register to the value stored in the CRC_INIT register, otherwise, reset Data Register to its default value.

Reference Manual to LL API cross reference:

- CR RESET LL_CRC_ResetCRCCalculationUnit

LL_CRC_FeedData32

Function name

__STATIC_INLINE void LL_CRC_FeedData32 (CRC_TypeDef * CRCx, uint32_t InData)

Function description

Write given 32-bit data to the CRC calculator.

Parameters

- **CRCx:** CRC Instance
- **InData:** value to be provided to CRC calculator between between Min_Data=0 and Max_Data=0xFFFFFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR DR LL_CRC_FeedData32

LL_CRC_ReadData32

Function name

__STATIC_INLINE uint32_t LL_CRC_ReadData32 (CRC_TypeDef * CRCx)

Function description

Return current CRC calculation result.

Parameters

- **CRCx:** CRC Instance

Return values

- **Current:** CRC calculation result as stored in CRC_DR register (32 bits).

Reference Manual to LL API cross reference:

- DR DR LL_CRC_ReadData32

LL_CRC_Read_IDR

Function name

__STATIC_INLINE uint32_t LL_CRC_Read_IDR (CRC_TypeDef * CRCx)

Function description

Return data stored in the Independent Data(IDR) register.

Parameters

- **CRCx:** CRC Instance

Return values

- **Value:** stored in CRC_IDR register (General-purpose 8-bit data register).

Notes

- This register can be used as a temporary storage location for one byte.

Reference Manual to LL API cross reference:

- IDR IDR LL_CRC_Read_IDR

LL_CRC_Write_IDR

Function name

__STATIC_INLINE void LL_CRC_Write_IDR (CRC_TypeDef * CRCx, uint32_t InData)

Function description

Store data in the Independent Data(IDR) register.

Parameters

- **CRCx:** CRC Instance
- **InData:** value to be stored in CRC_IDR register (8-bit) between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Notes

- This register can be used as a temporary storage location for one byte.

Reference Manual to LL API cross reference:

- IDR IDR LL_CRC_Write_IDR

LL_CRC_DeInit

Function name

ErrorStatus LL_CRC_DeInit (CRC_TypeDef * CRCx)

Function description

De-initialize CRC registers (Registers restored to their default values).

Parameters

- **CRCx:** CRC Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: CRC registers are de-initialized
  - ERROR: CRC registers are not de-initialized

76.2 CRC Firmware driver defines

The following section lists the various define and macros of the module.

76.2.1 CRC

CRC

Common Write and read registers Macros

LL_CRC_WriteReg

Description:

- Write a value in CRC register.

Parameters:

- __INSTANCE__: CRC Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_CRC_ReadReg

Description:

- Read a value in CRC register.

Parameters:

- __INSTANCE__: CRC Instance
- __REG__: Register to be read

Return value:

- Register: value

77 LL DAC Generic Driver

77.1 DAC Firmware driver registers structures

77.1.1 LL_DAC_InitTypeDef

LL_DAC_InitTypeDef is defined in the stm32f4xx_ll_dac.h

Data Fields

- ***uint32_t TriggerSource***
- ***uint32_t WaveAutoGeneration***
- ***uint32_t WaveAutoGenerationConfig***
- ***uint32_t OutputBuffer***

Field Documentation

- ***uint32_t LL_DAC_InitTypeDef::TriggerSource***

  Set the conversion trigger source for the selected DAC channel: internal (SW start) or from external IP (timer event, external interrupt line). This parameter can be a value of ***DAC_LL_EC_TRIGGER_SOURCE***This feature can be modified afterwards using unitary function **LL_DAC_SetTriggerSource()**.
- ***uint32_t LL_DAC_InitTypeDef::WaveAutoGeneration***

  Set the waveform automatic generation mode for the selected DAC channel. This parameter can be a value of ***DAC_LL_EC_WAVE_AUTO_GENERATION_MODE***This feature can be modified afterwards using unitary function **LL_DAC_SetWaveAutoGeneration()**.
- ***uint32_t LL_DAC_InitTypeDef::WaveAutoGenerationConfig***

  Set the waveform automatic generation mode for the selected DAC channel. If waveform automatic generation mode is set to noise, this parameter can be a value of ***DAC_LL_EC_WAVE_NOISE_LFSR_UNMASK_BITS*** If waveform automatic generation mode is set to triangle, this parameter can be a value of ***DAC_LL_EC_WAVE_TRIANGLE_AMPLITUDE***

  Note:

  – If waveform automatic generation mode is disabled, this parameter is discarded.

  This feature can be modified afterwards using unitary function **LL_DAC_SetWaveNoiseLFSR()** or **LL_DAC_SetWaveTriangleAmplitude()**, depending on the wave automatic generation selected.
- ***uint32_t LL_DAC_InitTypeDef::OutputBuffer***

  Set the output buffer for the selected DAC channel. This parameter can be a value of ***DAC_LL_EC_OUTPUT_BUFFER***This feature can be modified afterwards using unitary function **LL_DAC_SetOutputBuffer()**.

77.2 DAC Firmware driver API description

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

77.2.1 Detailed description of functions

LL_DAC_SetTriggerSource

Function name

__STATIC_INLINE void LL_DAC_SetTriggerSource (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t TriggerSource)

Function description

Set the conversion trigger source for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **TriggerSource:** This parameter can be one of the following values:
  - LL_DAC_TRIG_SOFTWARE
  - LL_DAC_TRIG_EXT_TIM8_TRGO
  - LL_DAC_TRIG_EXT_TIM7_TRGO
  - LL_DAC_TRIG_EXT_TIM6_TRGO
  - LL_DAC_TRIG_EXT_TIM5_TRGO
  - LL_DAC_TRIG_EXT_TIM4_TRGO
  - LL_DAC_TRIG_EXT_TIM2_TRGO
  - LL_DAC_TRIG_EXT_EXTI_LINE9

Return values

- **None:**

Notes

- For conversion trigger source to be effective, DAC trigger must be enabled using function LL_DAC_EnableTrigger().
- To set conversion trigger source, DAC channel must be disabled. Otherwise, the setting is discarded.
- Availability of parameters of trigger sources from timer depends on timers availability on the selected device.

Reference Manual to LL API cross reference:

- CR TSEL1 LL_DAC_SetTriggerSource
- CR TSEL2 LL_DAC_SetTriggerSource

LL_DAC_GetTriggerSource

Function name

__STATIC_INLINE uint32_t LL_DAC_GetTriggerSource (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Get the conversion trigger source for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **Returned:** value can be one of the following values:
  - LL_DAC_TRIG_SOFTWARE
  - LL_DAC_TRIG_EXT_TIM8_TRGO
  - LL_DAC_TRIG_EXT_TIM7_TRGO
  - LL_DAC_TRIG_EXT_TIM6_TRGO
  - LL_DAC_TRIG_EXT_TIM5_TRGO
  - LL_DAC_TRIG_EXT_TIM4_TRGO
  - LL_DAC_TRIG_EXT_TIM2_TRGO
  - LL_DAC_TRIG_EXT_EXTI_LINE9

Notes

- For conversion trigger source to be effective, DAC trigger must be enabled using function LL_DAC_EnableTrigger().
- Availability of parameters of trigger sources from timer depends on timers availability on the selected device.

Reference Manual to LL API cross reference:

- CR TSEL1 LL_DAC_GetTriggerSource
- CR TSEL2 LL_DAC_GetTriggerSource

LL_DAC_SetWaveAutoGeneration

Function name

__STATIC_INLINE void LL_DAC_SetWaveAutoGeneration (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t WaveAutoGeneration)

Function description

Set the waveform automatic generation mode for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **WaveAutoGeneration:** This parameter can be one of the following values:
  - LL_DAC_WAVE_AUTO_GENERATION_NONE
  - LL_DAC_WAVE_AUTO_GENERATION_NOISE
  - LL_DAC_WAVE_AUTO_GENERATION_TRIANGLE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR WAVE1 LL_DAC_SetWaveAutoGeneration
- CR WAVE2 LL_DAC_SetWaveAutoGeneration

LL_DAC_GetWaveAutoGeneration

Function name

__STATIC_INLINE uint32_t LL_DAC_GetWaveAutoGeneration (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Get the waveform automatic generation mode for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **Returned:** value can be one of the following values:
  - LL_DAC_WAVE_AUTO_GENERATION_NONE
  - LL_DAC_WAVE_AUTO_GENERATION_NOISE
  - LL_DAC_WAVE_AUTO_GENERATION_TRIANGLE

Reference Manual to LL API cross reference:

- CR WAVE1 LL_DAC_GetWaveAutoGeneration
- CR WAVE2 LL_DAC_GetWaveAutoGeneration

LL_DAC_SetWaveNoiseLFSR

Function name

__STATIC_INLINE void LL_DAC_SetWaveNoiseLFSR (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t NoiseLFSRMask)

Function description

Set the noise waveform generation for the selected DAC channel: Noise mode and parameters LFSR (linear feedback shift register).

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **NoiseLFSRMask:** This parameter can be one of the following values:
  - LL_DAC_NOISE_LFSR_UNMASK_BIT0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS1_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS2_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS3_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS4_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS5_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS6_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS7_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS8_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS9_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS10_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS11_0

Return values

- **None:**

Notes

- For wave generation to be effective, DAC channel wave generation mode must be enabled using function LL_DAC_SetWaveAutoGeneration().
- This setting can be set when the selected DAC channel is disabled (otherwise, the setting operation is ignored).

Reference Manual to LL API cross reference:

- CR MAMP1 LL_DAC_SetWaveNoiseLFSR
- CR MAMP2 LL_DAC_SetWaveNoiseLFSR

LL_DAC_GetWaveNoiseLFSR

Function name

__STATIC_INLINE uint32_t LL_DAC_GetWaveNoiseLFSR (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Set the noise waveform generation for the selected DAC channel: Noise mode and parameters LFSR (linear feedback shift register).

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **Returned:** value can be one of the following values:
  - LL_DAC_NOISE_LFSR_UNMASK_BIT0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS1_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS2_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS3_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS4_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS5_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS6_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS7_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS8_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS9_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS10_0
  - LL_DAC_NOISE_LFSR_UNMASK_BITS11_0

Reference Manual to LL API cross reference:

- CR MAMP1 LL_DAC_GetWaveNoiseLFSR
- CR MAMP2 LL_DAC_GetWaveNoiseLFSR

LL_DAC_SetWaveTriangleAmplitude

Function name

__STATIC_INLINE void LL_DAC_SetWaveTriangleAmplitude (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t TriangleAmplitude)

Function description

Set the triangle waveform generation for the selected DAC channel: triangle mode and amplitude.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **TriangleAmplitude:** This parameter can be one of the following values:
  - LL_DAC_TRIANGLE_AMPLITUDE_1
  - LL_DAC_TRIANGLE_AMPLITUDE_3
  - LL_DAC_TRIANGLE_AMPLITUDE_7
  - LL_DAC_TRIANGLE_AMPLITUDE_15
  - LL_DAC_TRIANGLE_AMPLITUDE_31
  - LL_DAC_TRIANGLE_AMPLITUDE_63
  - LL_DAC_TRIANGLE_AMPLITUDE_127
  - LL_DAC_TRIANGLE_AMPLITUDE_255
  - LL_DAC_TRIANGLE_AMPLITUDE_511
  - LL_DAC_TRIANGLE_AMPLITUDE_1023
  - LL_DAC_TRIANGLE_AMPLITUDE_2047
  - LL_DAC_TRIANGLE_AMPLITUDE_4095

Return values

- **None:**

Notes

- For wave generation to be effective, DAC channel wave generation mode must be enabled using function LL_DAC_SetWaveAutoGeneration().
- This setting can be set when the selected DAC channel is disabled (otherwise, the setting operation is ignored).

Reference Manual to LL API cross reference:

- CR MAMP1 LL_DAC_SetWaveTriangleAmplitude
- CR MAMP2 LL_DAC_SetWaveTriangleAmplitude

LL_DAC_GetWaveTriangleAmplitude

Function name

__STATIC_INLINE uint32_t LL_DAC_GetWaveTriangleAmplitude (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Set the triangle waveform generation for the selected DAC channel: triangle mode and amplitude.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **Returned:** value can be one of the following values:
  - LL_DAC_TRIANGLE_AMPLITUDE_1
  - LL_DAC_TRIANGLE_AMPLITUDE_3
  - LL_DAC_TRIANGLE_AMPLITUDE_7
  - LL_DAC_TRIANGLE_AMPLITUDE_15
  - LL_DAC_TRIANGLE_AMPLITUDE_31
  - LL_DAC_TRIANGLE_AMPLITUDE_63
  - LL_DAC_TRIANGLE_AMPLITUDE_127
  - LL_DAC_TRIANGLE_AMPLITUDE_255
  - LL_DAC_TRIANGLE_AMPLITUDE_511
  - LL_DAC_TRIANGLE_AMPLITUDE_1023
  - LL_DAC_TRIANGLE_AMPLITUDE_2047
  - LL_DAC_TRIANGLE_AMPLITUDE_4095

Reference Manual to LL API cross reference:

- CR MAMP1 LL_DAC_GetWaveTriangleAmplitude
- CR MAMP2 LL_DAC_GetWaveTriangleAmplitude

LL_DAC_SetOutputBuffer

Function name

__STATIC_INLINE void LL_DAC_SetOutputBuffer (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t OutputBuffer)

Function description

Set the output buffer for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **OutputBuffer:** This parameter can be one of the following values:
  - LL_DAC_OUTPUT_BUFFER_ENABLE
  - LL_DAC_OUTPUT_BUFFER_DISABLE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR BOFF1 LL_DAC_SetOutputBuffer
- CR BOFF2 LL_DAC_SetOutputBuffer

LL_DAC_GetOutputBuffer

Function name

__STATIC_INLINE uint32_t LL_DAC_GetOutputBuffer (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Get the output buffer state for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **Returned:** value can be one of the following values:
  - LL_DAC_OUTPUT_BUFFER_ENABLE
  - LL_DAC_OUTPUT_BUFFER_DISABLE

Reference Manual to LL API cross reference:

- CR BOFF1 LL_DAC_GetOutputBuffer
- CR BOFF2 LL_DAC_GetOutputBuffer

LL_DAC_EnableDMAReq

Function name

__STATIC_INLINE void LL_DAC_EnableDMAReq (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Enable DAC DMA transfer request of the selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **None:**

Notes

- To configure DMA source address (peripheral address), use function LL_DAC_DMA_GetRegAddr().

Reference Manual to LL API cross reference:

- CR DMAEN1 LL_DAC_EnableDMAReq
- CR DMAEN2 LL_DAC_EnableDMAReq

LL_DAC_DisableDMAReq

Function name

__STATIC_INLINE void LL_DAC_DisableDMAReq (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Disable DAC DMA transfer request of the selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **None:**

Notes

- To configure DMA source address (peripheral address), use function LL_DAC_DMA_GetRegAddr().

Reference Manual to LL API cross reference:

- CR DMAEN1 LL_DAC_DisableDMAReq
- CR DMAEN2 LL_DAC_DisableDMAReq

LL_DAC_IsDMAReqEnabled

Function name

__STATIC_INLINE uint32_t LL_DAC_IsDMAReqEnabled (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Get DAC DMA transfer request state of the selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR DMAEN1 LL_DAC_IsDMAReqEnabled
- CR DMAEN2 LL_DAC_IsDMAReqEnabled

LL_DAC_DMA_GetRegAddr

Function name

__STATIC_INLINE uint32_t LL_DAC_DMA_GetRegAddr (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Register)

Function description

Function to help to configure DMA transfer to DAC: retrieve the DAC register address from DAC instance and a list of DAC registers intended to be used (most commonly) with DMA transfer.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **Register:** This parameter can be one of the following values:
  - LL_DAC_DMA_REG_DATA_12BITS_RIGHT_ALIGNED
  - LL_DAC_DMA_REG_DATA_12BITS_LEFT_ALIGNED
  - LL_DAC_DMA_REG_DATA_8BITS_RIGHT_ALIGNED

Return values

- **DAC:** register address

Notes

- These DAC registers are data holding registers: when DAC conversion is requested, DAC generates a DMA transfer request to have data available in DAC data holding registers.
- This macro is intended to be used with LL DMA driver, refer to function "LL_DMA_ConfigAddresses()". Example: LL_DMA_ConfigAddresses(DMA1, LL_DMA_CHANNEL_1, (uint32_t)&< array or variable >, LL_DAC_DMA_GetRegAddr(DAC1, LL_DAC_CHANNEL_1, LL_DAC_DMA_REG_DATA_12BITS_RIGHT_ALIGNED), LL_DMA_DIRECTION_MEMORY_TO_PERIPH);

Reference Manual to LL API cross reference:

- DHR12R1 DACC1DHR LL_DAC_DMA_GetRegAddr
- DHR12L1 DACC1DHR LL_DAC_DMA_GetRegAddr
- DHR8R1 DACC1DHR LL_DAC_DMA_GetRegAddr
- DHR12R2 DACC2DHR LL_DAC_DMA_GetRegAddr
- DHR12L2 DACC2DHR LL_DAC_DMA_GetRegAddr
- DHR8R2 DACC2DHR LL_DAC_DMA_GetRegAddr

LL_DAC_Enable

Function name

__STATIC_INLINE void LL_DAC_Enable (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Enable DAC selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **None:**

Notes

- After enable from off state, DAC channel requires a delay for output voltage to reach accuracy +/- 1 LSB. Refer to device datasheet, parameter "tWAKEUP".

Reference Manual to LL API cross reference:

- CR EN1 LL_DAC_Enable
- CR EN2 LL_DAC_Enable

LL_DAC_Disable

Function name

__STATIC_INLINE void LL_DAC_Disable (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Disable DAC selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR EN1 LL_DAC_Disable
- CR EN2 LL_DAC_Disable

LL_DAC_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_DAC_IsEnabled (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Get DAC enable state of the selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR EN1 LL_DAC_IsEnabled
- CR EN2 LL_DAC_IsEnabled

LL_DAC_EnableTrigger

Function name

__STATIC_INLINE void LL_DAC_EnableTrigger (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Enable DAC trigger of the selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **None:**

Notes

- - If DAC trigger is disabled, DAC conversion is performed automatically once the data holding register is updated, using functions "LL_DAC_ConvertData{8; 12}{Right; Left} Aligned()": LL_DAC_ConvertData12RightAligned(), ... If DAC trigger is enabled, DAC conversion is performed only when a hardware of software trigger event is occurring. Select trigger source using function LL_DAC_SetTriggerSource().

Reference Manual to LL API cross reference:

- CR TEN1 LL_DAC_EnableTrigger
- CR TEN2 LL_DAC_EnableTrigger

LL_DAC_DisableTrigger

Function name

__STATIC_INLINE void LL_DAC_DisableTrigger (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Disable DAC trigger of the selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TEN1 LL_DAC_DisableTrigger
- CR TEN2 LL_DAC_DisableTrigger

LL_DAC_IsTriggerEnabled

Function name

__STATIC_INLINE uint32_t LL_DAC_IsTriggerEnabled (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Get DAC trigger state of the selected channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR TEN1 LL_DAC_IsTriggerEnabled
- CR TEN2 LL_DAC_IsTriggerEnabled

LL_DAC_TrigSWConversion

Function name

__STATIC_INLINE void LL_DAC_TrigSWConversion (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Trig DAC conversion by software for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can a combination of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **None:**

Notes

- Preliminarily, DAC trigger must be set to software trigger using function LL_DAC_SetTriggerSource() with parameter "LL_DAC_TRIGGER_SOFTWARE". and DAC trigger must be enabled using function LL_DAC_EnableTrigger().
- For devices featuring DAC with 2 channels: this function can perform a SW start of both DAC channels simultaneously. Two channels can be selected as parameter. Example: (LL_DAC_CHANNEL_1 | LL_DAC_CHANNEL_2)

Reference Manual to LL API cross reference:

- SWTRIGR SWTRIG1 LL_DAC_TrigSWConversion
- SWTRIGR SWTRIG2 LL_DAC_TrigSWConversion

LL_DAC_ConvertData12RightAligned

Function name

__STATIC_INLINE void LL_DAC_ConvertData12RightAligned (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)

Function description

Set the data to be loaded in the data holding register in format 12 bits left alignment (LSB aligned on bit 0), for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **Data:** Value between Min_Data=0x000 and Max_Data=0xFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DHR12R1 DACC1DHR LL_DAC_ConvertData12RightAligned
- DHR12R2 DACC2DHR LL_DAC_ConvertData12RightAligned

LL_DAC_ConvertData12LeftAligned

Function name

__STATIC_INLINE void LL_DAC_ConvertData12LeftAligned (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)

Function description

Set the data to be loaded in the data holding register in format 12 bits left alignment (MSB aligned on bit 15), for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **Data:** Value between Min_Data=0x000 and Max_Data=0xFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DHR12L1 DACC1DHR LL_DAC_ConvertData12LeftAligned
- DHR12L2 DACC2DHR LL_DAC_ConvertData12LeftAligned

LL_DAC_ConvertData8RightAligned

Function name

__STATIC_INLINE void LL_DAC_ConvertData8RightAligned (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)

Function description

Set the data to be loaded in the data holding register in format 8 bits left alignment (LSB aligned on bit 0), for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **Data:** Value between Min_Data=0x00 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DHR8R1 DACC1DHR LL_DAC_ConvertData8RightAligned
- DHR8R2 DACC2DHR LL_DAC_ConvertData8RightAligned

LL_DAC_ConvertDualData12RightAligned

Function name

__STATIC_INLINE void LL_DAC_ConvertDualData12RightAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)

Function description

Set the data to be loaded in the data holding register in format 12 bits left alignment (LSB aligned on bit 0), for both DAC channels.

Parameters

- **DACx:** DAC instance
- **DataChannel1:** Value between Min_Data=0x000 and Max_Data=0xFFF
- **DataChannel2:** Value between Min_Data=0x000 and Max_Data=0xFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DHR12RD DACC1DHR LL_DAC_ConvertDualData12RightAligned
- DHR12RD DACC2DHR LL_DAC_ConvertDualData12RightAligned

LL_DAC_ConvertDualData12LeftAligned

Function name

__STATIC_INLINE void LL_DAC_ConvertDualData12LeftAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)

Function description

Set the data to be loaded in the data holding register in format 12 bits left alignment (MSB aligned on bit 15), for both DAC channels.

Parameters

- **DACx:** DAC instance
- **DataChannel1:** Value between Min_Data=0x000 and Max_Data=0xFFF
- **DataChannel2:** Value between Min_Data=0x000 and Max_Data=0xFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DHR12LD DACC1DHR LL_DAC_ConvertDualData12LeftAligned
- DHR12LD DACC2DHR LL_DAC_ConvertDualData12LeftAligned

LL_DAC_ConvertDualData8RightAligned

Function name

__STATIC_INLINE void LL_DAC_ConvertDualData8RightAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)

Function description

Set the data to be loaded in the data holding register in format 8 bits left alignment (LSB aligned on bit 0), for both DAC channels.

Parameters

- **DACx:** DAC instance
- **DataChannel1:** Value between Min_Data=0x00 and Max_Data=0xFF
- **DataChannel2:** Value between Min_Data=0x00 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DHR8RD DACC1DHR LL_DAC_ConvertDualData8RightAligned
- DHR8RD DACC2DHR LL_DAC_ConvertDualData8RightAligned

LL_DAC_RetrieveOutputData

Function name

__STATIC_INLINE uint32_t LL_DAC_RetrieveOutputData (DAC_TypeDef * DACx, uint32_t DAC_Channel)

Function description

Retrieve output data currently generated for the selected DAC channel.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.

Return values

- **Value:** between Min_Data=0x000 and Max_Data=0xFFF

Notes

- Whatever alignment and resolution settings (using functions "LL_DAC_ConvertData{8; 12}{Right; Left} Aligned()": LL_DAC_ConvertData12RightAligned(), ...), output data format is 12 bits right aligned (LSB aligned on bit 0).

Reference Manual to LL API cross reference:

- DOR1 DACC1DOR LL_DAC_RetrieveOutputData
- DOR2 DACC2DOR LL_DAC_RetrieveOutputData

LL_DAC_IsActiveFlag_DMAUDR1

Function name

__STATIC_INLINE uint32_t LL_DAC_IsActiveFlag_DMAUDR1 (DAC_TypeDef * DACx)

Function description

Get DAC underrun flag for DAC channel 1.

Parameters

- **DACx:** DAC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR DMAUDR1 LL_DAC_IsActiveFlag_DMAUDR1

LL_DAC_IsActiveFlag_DMAUDR2

Function name

__STATIC_INLINE uint32_t LL_DAC_IsActiveFlag_DMAUDR2 (DAC_TypeDef * DACx)

Function description

Get DAC underrun flag for DAC channel 2.

Parameters

- **DACx:** DAC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR DMAUDR2 LL_DAC_IsActiveFlag_DMAUDR2

LL_DAC_ClearFlag_DMAUDR1

Function name

__STATIC_INLINE void LL_DAC_ClearFlag_DMAUDR1 (DAC_TypeDef * DACx)

Function description

Clear DAC underrun flag for DAC channel 1.

Parameters

- **DACx:** DAC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR DMAUDR1 LL_DAC_ClearFlag_DMAUDR1

LL_DAC_ClearFlag_DMAUDR2

Function name

__STATIC_INLINE void LL_DAC_ClearFlag_DMAUDR2 (DAC_TypeDef * DACx)

Function description

Clear DAC underrun flag for DAC channel 2.

Parameters

- **DACx:** DAC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR DMAUDR2 LL_DAC_ClearFlag_DMAUDR2

LL_DAC_EnableIT_DMAUDR1

Function name

__STATIC_INLINE void LL_DAC_EnableIT_DMAUDR1 (DAC_TypeDef * DACx)

Function description

Enable DMA underrun interrupt for DAC channel 1.

Parameters

- **DACx:** DAC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DMAUDRIE1 LL_DAC_EnableIT_DMAUDR1

LL_DAC_EnableIT_DMAUDR2

Function name

__STATIC_INLINE void LL_DAC_EnableIT_DMAUDR2 (DAC_TypeDef * DACx)

Function description

Enable DMA underrun interrupt for DAC channel 2.

Parameters

- **DACx:** DAC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DMAUDRIE2 LL_DAC_EnableIT_DMAUDR2

LL_DAC_DisableIT_DMAUDR1

Function name

__STATIC_INLINE void LL_DAC_DisableIT_DMAUDR1 (DAC_TypeDef * DACx)

Function description

Disable DMA underrun interrupt for DAC channel 1.

Parameters

- **DACx:** DAC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DMAUDRIE1 LL_DAC_DisableIT_DMAUDR1

LL_DAC_DisableIT_DMAUDR2

Function name

__STATIC_INLINE void LL_DAC_DisableIT_DMAUDR2 (DAC_TypeDef * DACx)

Function description

Disable DMA underrun interrupt for DAC channel 2.

Parameters

- **DACx:** DAC instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DMAUDRIE2 LL_DAC_DisableIT_DMAUDR2

LL_DAC_IsEnabledIT_DMAUDR1

Function name

__STATIC_INLINE uint32_t LL_DAC_IsEnabledIT_DMAUDR1 (DAC_TypeDef * DACx)

Function description

Get DMA underrun interrupt for DAC channel 1.

Parameters

- **DACx:** DAC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR DMAUDRIE1 LL_DAC_IsEnabledIT_DMAUDR1

LL_DAC_IsEnabledIT_DMAUDR2

Function name

__STATIC_INLINE uint32_t LL_DAC_IsEnabledIT_DMAUDR2 (DAC_TypeDef * DACx)

Function description

Get DMA underrun interrupt for DAC channel 2.

Parameters

- **DACx:** DAC instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR DMAUDRIE2 LL_DAC_IsEnabledIT_DMAUDR2

LL_DAC_DeInit

Function name

ErrorStatus LL_DAC_DeInit (DAC_TypeDef * DACx)

Function description

De-initialize registers of the selected DAC instance to their default reset values.

Parameters

- **DACx:** DAC instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: DAC registers are de-initialized
  - ERROR: not applicable

LL_DAC_Init

Function name

ErrorStatus LL_DAC_Init (DAC_TypeDef * DACx, uint32_t DAC_Channel, LL_DAC_InitTypeDef * DAC_InitStruct)

Function description

Initialize some features of DAC instance.

Parameters

- **DACx:** DAC instance
- **DAC_Channel:** This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

  (1) On this STM32 serie, parameter not available on all devices. Refer to device datasheet for channels availability.
- **DAC_InitStruct:** Pointer to a LL_DAC_InitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: DAC registers are initialized
  - ERROR: DAC registers are not initialized

Notes

- The setting of these parameters by function LL_DAC_Init() is conditioned to DAC state: DAC instance must be disabled.

LL_DAC_StructInit

Function name

void LL_DAC_StructInit (LL_DAC_InitTypeDef * DAC_InitStruct)

Function description

Set each LL_DAC_InitTypeDef field to default value.

Parameters

- **DAC_InitStruct:** pointer to a LL_DAC_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

77.3 DAC Firmware driver defines

The following section lists the various define and macros of the module.

77.3.1 DAC

DAC

DAC channels

LL_DAC_CHANNEL_1

DAC channel 1

LL_DAC_CHANNEL_2

DAC channel 2

DAC flags

LL_DAC_FLAG_DMAUDR1

DAC channel 1 flag DMA underrun

LL_DAC_FLAG_DMAUDR2

DAC channel 2 flag DMA underrun

Definitions of DAC hardware constraints delays

LL_DAC_DELAY_STARTUP_VOLTAGE_SETTLING_US

Delay for DAC channel voltage settling time from DAC channel startup (transition from disable to enable)

LL_DAC_DELAY_VOLTAGE_SETTLING_US

Delay for DAC channel voltage settling time

DAC interruptions

LL_DAC_IT_DMAUDRIE1

DAC channel 1 interruption DMA underrun

LL_DAC_IT_DMAUDRIE2

DAC channel 2 interruption DMA underrun

DAC channel output buffer

LL_DAC_OUTPUT_BUFFER_ENABLE

The selected DAC channel output is buffered: higher drive current capability, but also higher current consumption

LL_DAC_OUTPUT_BUFFER_DISABLE

The selected DAC channel output is not buffered: lower drive current capability, but also lower current consumption

DAC registers compliant with specific purpose

LL_DAC_DMA_REG_DATA_12BITS_RIGHT_ALIGNED

DAC channel data holding register 12 bits right aligned

LL_DAC_DMA_REG_DATA_12BITS_LEFT_ALIGNED

DAC channel data holding register 12 bits left aligned

LL_DAC_DMA_REG_DATA_8BITS_RIGHT_ALIGNED

DAC channel data holding register 8 bits right aligned

DAC channel output resolution

LL_DAC_RESOLUTION_12B

DAC channel resolution 12 bits

LL_DAC_RESOLUTION_8B

DAC channel resolution 8 bits

DAC trigger source

LL_DAC_TRIG_SOFTWARE

DAC channel conversion trigger internal (SW start)

LL_DAC_TRIG_EXT_TIM2_TRGO

DAC channel conversion trigger from external IP: TIM2 TRGO.

LL_DAC_TRIG_EXT_TIM8_TRGO

DAC channel conversion trigger from external IP: TIM8 TRGO.

LL_DAC_TRIG_EXT_TIM4_TRGO

DAC channel conversion trigger from external IP: TIM4 TRGO.

LL_DAC_TRIG_EXT_TIM6_TRGO

DAC channel conversion trigger from external IP: TIM6 TRGO.

LL_DAC_TRIG_EXT_TIM7_TRGO

DAC channel conversion trigger from external IP: TIM7 TRGO.

LL_DAC_TRIG_EXT_TIM5_TRGO

DAC channel conversion trigger from external IP: TIM5 TRGO.

LL_DAC_TRIG_EXT_EXTI_LINE9

DAC channel conversion trigger from external IP: external interrupt line 9.

DAC waveform automatic generation mode

LL_DAC_WAVE_AUTO_GENERATION_NONE

DAC channel wave auto generation mode disabled.

LL_DAC_WAVE_AUTO_GENERATION_NOISE

DAC channel wave auto generation mode enabled, set generated noise waveform.

LL_DAC_WAVE_AUTO_GENERATION_TRIANGLE

DAC channel wave auto generation mode enabled, set generated triangle waveform.

DAC wave generation - Noise LFSR unmask bits

LL_DAC_NOISE_LFSR_UNMASK_BIT0

Noise wave generation, unmask LFSR bit0, for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS1_0

Noise wave generation, unmask LFSR bits[1:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS2_0

Noise wave generation, unmask LFSR bits[2:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS3_0

Noise wave generation, unmask LFSR bits[3:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS4_0

Noise wave generation, unmask LFSR bits[4:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS5_0

Noise wave generation, unmask LFSR bits[5:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS6_0

Noise wave generation, unmask LFSR bits[6:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS7_0

Noise wave generation, unmask LFSR bits[7:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS8_0

Noise wave generation, unmask LFSR bits[8:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS9_0

Noise wave generation, unmask LFSR bits[9:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS10_0

Noise wave generation, unmask LFSR bits[10:0], for the selected DAC channel

LL_DAC_NOISE_LFSR_UNMASK_BITS11_0

Noise wave generation, unmask LFSR bits[11:0], for the selected DAC channel

DAC wave generation - Triangle amplitude

LL_DAC_TRIANGLE_AMPLITUDE_1

Triangle wave generation, amplitude of 1 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_3

Triangle wave generation, amplitude of 3 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_7

Triangle wave generation, amplitude of 7 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_15

Triangle wave generation, amplitude of 15 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_31

Triangle wave generation, amplitude of 31 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_63

Triangle wave generation, amplitude of 63 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_127

Triangle wave generation, amplitude of 127 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_255

Triangle wave generation, amplitude of 255 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_511

Triangle wave generation, amplitude of 512 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_1023

Triangle wave generation, amplitude of 1023 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_2047

Triangle wave generation, amplitude of 2047 LSB of DAC output range, for the selected DAC channel

LL_DAC_TRIANGLE_AMPLITUDE_4095

Triangle wave generation, amplitude of 4095 LSB of DAC output range, for the selected DAC channel

DAC helper macro

__LL_DAC_CHANNEL_TO_DECIMAL_NB

Description:

- Helper macro to get DAC channel number in decimal format from literals LL_DAC_CHANNEL_x.

Parameters:

- __CHANNEL__: This parameter can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

Return value:

- 1...2: (value "2" depending on DAC channel 2 availability)

Notes:

- The input can be a value from functions where a channel number is returned.

__LL_DAC_DECIMAL_NB_TO_CHANNEL

Description:

- Helper macro to get DAC channel in literal format LL_DAC_CHANNEL_x from number in decimal format.

Parameters:

- __DECIMAL_NB__: 1...2 (value "2" depending on DAC channel 2 availability)

Return value:

- Returned: value can be one of the following values:
  - LL_DAC_CHANNEL_1
  - LL_DAC_CHANNEL_2 (1)

Notes:

- If the input parameter does not correspond to a DAC channel, this macro returns value '0'.

__LL_DAC_DIGITAL_SCALE

Description:

- Helper macro to define the DAC conversion data full-scale digital value corresponding to the selected DAC resolution.

Parameters:

- __DAC_RESOLUTION__: This parameter can be one of the following values:
  - LL_DAC_RESOLUTION_12B
  - LL_DAC_RESOLUTION_8B

Return value:

- ADC: conversion data equivalent voltage value (unit: mVolt)

Notes:

- DAC conversion data full-scale corresponds to voltage range determined by analog voltage references Vref+ and Vref- (refer to reference manual).

__LL_DAC_CALC_VOLTAGE_TO_DATA

Description:

- Helper macro to calculate the DAC conversion data (unit: digital value) corresponding to a voltage (unit: mVolt).

Parameters:

- __VREFANALOG_VOLTAGE__: Analog reference voltage (unit mV)
- __DAC_VOLTAGE__: Voltage to be generated by DAC channel (unit: mVolt).
- __DAC_RESOLUTION__: This parameter can be one of the following values:
  - LL_DAC_RESOLUTION_12B
  - LL_DAC_RESOLUTION_8B

Return value:

- DAC: conversion data (unit: digital value)

Notes:

- This helper macro is intended to provide input data in voltage rather than digital value, to be used with LL DAC functions such as LL_DAC_ConvertData12RightAligned(). Analog reference voltage (Vref+) must be either known from user board environment or can be calculated using ADC measurement and ADC helper macro __LL_ADC_CALC_VREFANALOG_VOLTAGE().

Common write and read registers macros

LL_DAC_WriteReg

Description:

- Write a value in DAC register.

Parameters:

- __INSTANCE__: DAC Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_DAC_ReadReg

Description:

- Read a value in DAC register.

Parameters:

- __INSTANCE__: DAC Instance
- __REG__: Register to be read

Return value:

- Register: value

78 LL DMA2D Generic Driver

78.1 DMA2D Firmware driver registers structures

78.1.1 LL_DMA2D_InitTypeDef

LL_DMA2D_InitTypeDef is defined in the stm32f4xx_ll_dma2d.h

Data Fields

- ***uint32_t Mode***
- ***uint32_t ColorMode***
- ***uint32_t OutputBlue***
- ***uint32_t OutputGreen***
- ***uint32_t OutputRed***
- ***uint32_t OutputAlpha***
- ***uint32_t OutputMemoryAddress***
- ***uint32_t LineOffset***
- ***uint32_t NbrOfLines***
- ***uint32_t NbrOfPixelsPerLines***

Field Documentation

- ***uint32_t LL_DMA2D_InitTypeDef::Mode***

  Specifies the DMA2D transfer mode.
  - This parameter can be one value of ***DMA2D_LL_EC_MODE***.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetMode()**.
- ***uint32_t LL_DMA2D_InitTypeDef::ColorMode***

  Specifies the color format of the output image.
  - This parameter can be one value of ***DMA2D_LL_EC_OUTPUT_COLOR_MODE***.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColorMode()**.
- ***uint32_t LL_DMA2D_InitTypeDef::OutputBlue***

  Specifies the Blue value of the output image.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if ARGB8888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if RGB888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if RGB565 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if ARGB1555 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x0F if ARGB4444 color mode is selected.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColor()** or configuration function **LL_DMA2D_ConfigOutputColor()**.

- ***uint32_t LL_DMA2D_InitTypeDef::OutputGreen***

  Specifies the Green value of the output image.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if ARGB8888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if RGB888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x3F if RGB565 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if ARGB1555 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x0F if ARGB4444 color mode is selected.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColor()** or configuration function **LL_DMA2D_ConfigOutputColor()**.
- ***uint32_t LL_DMA2D_InitTypeDef::OutputRed***

  Specifies the Red value of the output image.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if ARGB8888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if RGB888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if RGB565 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if ARGB1555 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x0F if ARGB4444 color mode is selected.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColor()** or configuration function **LL_DMA2D_ConfigOutputColor()**.
- ***uint32_t LL_DMA2D_InitTypeDef::OutputAlpha***

  Specifies the Alpha channel of the output image.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if ARGB8888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x01 if ARGB1555 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x0F if ARGB4444 color mode is selected.
  - This parameter is not considered if RGB888 or RGB565 color mode is selected.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColor()** or configuration function **LL_DMA2D_ConfigOutputColor()**.
- ***uint32_t LL_DMA2D_InitTypeDef::OutputMemoryAddress***

  Specifies the memory address.
  - This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFFFFFF.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputMemAddr()**.
- ***uint32_t LL_DMA2D_InitTypeDef::LineOffset***

  Specifies the output line offset value.
  - This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0x3FFF.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetLineOffset()**.
- ***uint32_t LL_DMA2D_InitTypeDef::NbrOfLines***

  Specifies the number of lines of the area to be transferred.
  - This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetNbrOfLines()**.

- ***uint32_t LL_DMA2D_InitTypeDef::NbrOfPixelsPerLines***
  Specifies the number of pixels per lines of the area to be transfered.
  – This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0x3FFF.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetNbrOfPixelsPerLines()**.

78.1.2 LL_DMA2D_LayerCfgTypeDef

LL_DMA2D_LayerCfgTypeDef is defined in the stm32f4xx_ll_dma2d.h

Data Fields

- ***uint32_t MemoryAddress***
- ***uint32_t LineOffset***
- ***uint32_t ColorMode***
- ***uint32_t CLUTColorMode***
- ***uint32_t CLUTSize***
- ***uint32_t AlphaMode***
- ***uint32_t Alpha***
- ***uint32_t Blue***
- ***uint32_t Green***
- ***uint32_t Red***
- ***uint32_t CLUTMemoryAddress***

Field Documentation

- ***uint32_t LL_DMA2D_LayerCfgTypeDef::MemoryAddress***
  Specifies the foreground or background memory address.
  – This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFFFFFF.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetMemAddr()** for foreground layer,
  – **LL_DMA2D_BGND_SetMemAddr()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::LineOffset***
  Specifies the foreground or background line offset value.
  – This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0x3FFF.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetLineOffset()** for foreground layer,
  – **LL_DMA2D_BGND_SetLineOffset()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::ColorMode***
  Specifies the foreground or background color mode.
  – This parameter can be one value of ***DMA2D_LL_EC_INPUT_COLOR_MODE***.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetColorMode()** for foreground layer,
  – **LL_DMA2D_BGND_SetColorMode()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::CLUTColorMode***
  Specifies the foreground or background CLUT color mode.
  – This parameter can be one value of ***DMA2D_LL_EC_CLUT_COLOR_MODE***.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetCLUTColorMode()** for foreground layer,
  – **LL_DMA2D_BGND_SetCLUTColorMode()** for background layer.

- ***uint32_t LL_DMA2D_LayerCfgTypeDef::CLUTSize***
  Specifies the foreground or background CLUT size.
  – This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetCLUTSize()** for foreground layer,
  – **LL_DMA2D_BGND_SetCLUTSize()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::AlphaMode***
  Specifies the foreground or background alpha mode.
  – This parameter can be one value of ***DMA2D_LL_EC_ALPHA_MODE***.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetAlphaMode()** for foreground layer,
  – **LL_DMA2D_BGND_SetAlphaMode()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::Alpha***
  Specifies the foreground or background Alpha value.
  – This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetAlpha()** for foreground layer,
  – **LL_DMA2D_BGND_SetAlpha()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::Blue***
  Specifies the foreground or background Blue color value.
  – This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetBlueColor()** for foreground layer,
  – **LL_DMA2D_BGND_SetBlueColor()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::Green***
  Specifies the foreground or background Green color value.
  – This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetGreenColor()** for foreground layer,
  – **LL_DMA2D_BGND_SetGreenColor()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::Red***
  Specifies the foreground or background Red color value.
  – This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetRedColor()** for foreground layer,
  – **LL_DMA2D_BGND_SetRedColor()** for background layer.
- ***uint32_t LL_DMA2D_LayerCfgTypeDef::CLUTMemoryAddress***
  Specifies the foreground or background CLUT memory address.
  – This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFFFFFF.

  This parameter can be modified afterwards using unitary functions
  – **LL_DMA2D_FGND_SetCLUTMemAddr()** for foreground layer,
  – **LL_DMA2D_BGND_SetCLUTMemAddr()** for background layer.

78.1.3 LL_DMA2D_ColorTypeDef

LL_DMA2D_ColorTypeDef is defined in the stm32f4xx_ll_dma2d.h

Data Fields

- ***uint32_t ColorMode***
- ***uint32_t OutputBlue***

- ***uint32_t OutputGreen***
- ***uint32_t OutputRed***
- ***uint32_t OutputAlpha***

Field Documentation

- ***uint32_t LL_DMA2D_ColorTypeDef::ColorMode***

  Specifies the color format of the output image.
  - This parameter can be one value of ***DMA2D_LL_EC_OUTPUT_COLOR_MODE***.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColorMode()**.
- ***uint32_t LL_DMA2D_ColorTypeDef::OutputBlue***

  Specifies the Blue value of the output image.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if ARGB8888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if RGB888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if RGB565 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if ARGB1555 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x0F if ARGB4444 color mode is selected.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColor()** or configuration function **LL_DMA2D_ConfigOutputColor()**.
- ***uint32_t LL_DMA2D_ColorTypeDef::OutputGreen***

  Specifies the Green value of the output image.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if ARGB8888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if RGB888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x3F if RGB565 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if ARGB1555 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x0F if ARGB4444 color mode is selected.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColor()** or configuration function **LL_DMA2D_ConfigOutputColor()**.
- ***uint32_t LL_DMA2D_ColorTypeDef::OutputRed***

  Specifies the Red value of the output image.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if ARGB8888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if RGB888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if RGB565 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x1F if ARGB1555 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x0F if ARGB4444 color mode is selected.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColor()** or configuration function **LL_DMA2D_ConfigOutputColor()**.

- ***uint32_t LL_DMA2D_ColorTypeDef::OutputAlpha***

  Specifies the Alpha channel of the output image.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF if ARGB8888 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x01 if ARGB1555 color mode is selected.
  - This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x0F if ARGB4444 color mode is selected.
  - This parameter is not considered if RGB888 or RGB565 color mode is selected.

  This parameter can be modified afterwards using unitary function **LL_DMA2D_SetOutputColor()** or configuration function **LL_DMA2D_ConfigOutputColor()**.

78.2 DMA2D Firmware driver API description

The following section lists the various functions of the DMA2D library.

78.2.1 Detailed description of functions

LL_DMA2D_Start

Function name

__STATIC_INLINE void LL_DMA2D_Start (DMA2D_TypeDef * DMA2Dx)

Function description

Start a DMA2D transfer.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR START LL_DMA2D_Start

LL_DMA2D_IsTransferOngoing

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsTransferOngoing (DMA2D_TypeDef * DMA2Dx)

Function description

Indicate if a DMA2D transfer is ongoing.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR START LL_DMA2D_IsTransferOngoing

LL_DMA2D_Suspend

Function name

__STATIC_INLINE void LL_DMA2D_Suspend (DMA2D_TypeDef * DMA2Dx)

Function description

Suspend DMA2D transfer.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Notes

- This API can be used to suspend automatic foreground or background CLUT loading.

Reference Manual to LL API cross reference:

- CR SUSP LL_DMA2D_Suspend

LL_DMA2D_Resume

Function name

__STATIC_INLINE void LL_DMA2D_Resume (DMA2D_TypeDef * DMA2Dx)

Function description

Resume DMA2D transfer.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Notes

- This API can be used to resume automatic foreground or background CLUT loading.

Reference Manual to LL API cross reference:

- CR SUSP LL_DMA2D_Resume

LL_DMA2D_IsSuspended

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsSuspended (DMA2D_TypeDef * DMA2Dx)

Function description

Indicate if DMA2D transfer is suspended.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Notes

- This API can be used to indicate whether or not automatic foreground or background CLUT loading is suspended.

Reference Manual to LL API cross reference:

- CR SUSP LL_DMA2D_IsSuspended

LL_DMA2D_Abort

Function name

__STATIC_INLINE void LL_DMA2D_Abort (DMA2D_TypeDef * DMA2Dx)

Function description

Abort DMA2D transfer.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Notes

- This API can be used to abort automatic foreground or background CLUT loading.

Reference Manual to LL API cross reference:

- CR ABORT LL_DMA2D_Abort

LL_DMA2D_IsAborted

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsAborted (DMA2D_TypeDef * DMA2Dx)

Function description

Indicate if DMA2D transfer is aborted.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Notes

- This API can be used to indicate whether or not automatic foreground or background CLUT loading is aborted.

Reference Manual to LL API cross reference:

- CR ABORT LL_DMA2D_IsAborted

LL_DMA2D_SetMode

Function name

__STATIC_INLINE void LL_DMA2D_SetMode (DMA2D_TypeDef * DMA2Dx, uint32_t Mode)

Function description

Set DMA2D mode.

Parameters

- **DMA2Dx:** DMA2D Instance
- **Mode:** This parameter can be one of the following values:
  - LL_DMA2D_MODE_M2M
  - LL_DMA2D_MODE_M2M_PFC
  - LL_DMA2D_MODE_M2M_BLEND
  - LL_DMA2D_MODE_R2M

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR MODE LL_DMA2D_SetMode

LL_DMA2D_GetMode

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetMode (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D mode.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA2D_MODE_M2M
  - LL_DMA2D_MODE_M2M_PFC
  - LL_DMA2D_MODE_M2M_BLEND
  - LL_DMA2D_MODE_R2M

Reference Manual to LL API cross reference:

- CR MODE LL_DMA2D_GetMode

LL_DMA2D_SetOutputColorMode

Function name

__STATIC_INLINE void LL_DMA2D_SetOutputColorMode (DMA2D_TypeDef * DMA2Dx, uint32_t ColorMode)

Function description

Set DMA2D output color mode.

Parameters

- **DMA2Dx:** DMA2D Instance
- **ColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_OUTPUT_MODE_ARGB8888
  - LL_DMA2D_OUTPUT_MODE_RGB888
  - LL_DMA2D_OUTPUT_MODE_RGB565
  - LL_DMA2D_OUTPUT_MODE_ARGB1555
  - LL_DMA2D_OUTPUT_MODE_ARGB4444

Return values

- **None:**

Reference Manual to LL API cross reference:

- OPFCCR CM LL_DMA2D_SetOutputColorMode

LL_DMA2D_GetOutputColorMode

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetOutputColorMode (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D output color mode.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA2D_OUTPUT_MODE_ARGB8888
  - LL_DMA2D_OUTPUT_MODE_RGB888
  - LL_DMA2D_OUTPUT_MODE_RGB565
  - LL_DMA2D_OUTPUT_MODE_ARGB1555
  - LL_DMA2D_OUTPUT_MODE_ARGB4444

Reference Manual to LL API cross reference:

- OPFCCR CM LL_DMA2D_GetOutputColorMode

LL_DMA2D_SetLineOffset

Function name

__STATIC_INLINE void LL_DMA2D_SetLineOffset (DMA2D_TypeDef * DMA2Dx, uint32_t LineOffset)

Function description

Set DMA2D line offset, expressed on 14 bits ([13:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **LineOffset:** Value between Min_Data=0 and Max_Data=0x3FFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- OOR LO LL_DMA2D_SetLineOffset

LL_DMA2D_GetLineOffset

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetLineOffset (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D line offset, expressed on 14 bits ([13:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Line:** offset value between Min_Data=0 and Max_Data=0x3FFF

Reference Manual to LL API cross reference:

- OOR LO LL_DMA2D_GetLineOffset

LL_DMA2D_SetNbrOfPixelsPerLines

Function name

__STATIC_INLINE void LL_DMA2D_SetNbrOfPixelsPerLines (DMA2D_TypeDef * DMA2Dx, uint32_t NbrOfPixelsPerLines)

Function description

Set DMA2D number of pixels per lines, expressed on 14 bits ([13:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **NbrOfPixelsPerLines:** Value between Min_Data=0 and Max_Data=0x3FFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- NLR PL LL_DMA2D_SetNbrOfPixelsPerLines

LL_DMA2D_GetNbrOfPixelsPerLines

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetNbrOfPixelsPerLines (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D number of pixels per lines, expressed on 14 bits ([13:0] bits)

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Number:** of pixels per lines value between Min_Data=0 and Max_Data=0x3FFF

Reference Manual to LL API cross reference:

- NLR PL LL_DMA2D_GetNbrOfPixelsPerLines

LL_DMA2D_SetNbrOfLines

Function name

__STATIC_INLINE void LL_DMA2D_SetNbrOfLines (DMA2D_TypeDef * DMA2Dx, uint32_t NbrOfLines)

Function description

Set DMA2D number of lines, expressed on 16 bits ([15:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **NbrOfLines:** Value between Min_Data=0 and Max_Data=0xFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- NLR NL LL_DMA2D_SetNbrOfLines

LL_DMA2D_GetNbrOfLines

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetNbrOfLines (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D number of lines, expressed on 16 bits ([15:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Number:** of lines value between Min_Data=0 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- NLR NL LL_DMA2D_GetNbrOfLines

LL_DMA2D_SetOutputMemAddr

Function name

__STATIC_INLINE void LL_DMA2D_SetOutputMemAddr (DMA2D_TypeDef * DMA2Dx, uint32_t OutputMemoryAddress)

Function description

Set DMA2D output memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **OutputMemoryAddress:** Value between Min_Data=0 and Max_Data=0xFFFFFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- OMAR MA LL_DMA2D_SetOutputMemAddr

LL_DMA2D_GetOutputMemAddr

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetOutputMemAddr (DMA2D_TypeDef * DMA2Dx)

Function description

Get DMA2D output memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Output:** memory address value between Min_Data=0 and Max_Data=0xFFFFFFFF

Reference Manual to LL API cross reference:

- OMAR MA LL_DMA2D_GetOutputMemAddr

LL_DMA2D_SetOutputColor

Function name

__STATIC_INLINE void LL_DMA2D_SetOutputColor (DMA2D_TypeDef * DMA2Dx, uint32_t OutputColor)

Function description

Set DMA2D output color, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **OutputColor:** Value between Min_Data=0 and Max_Data=0xFFFFFFFF

Return values

- **None:**

Notes

- Output color format depends on output color mode, ARGB8888, RGB888, RGB565, ARGB1555 or ARGB4444.
- LL_DMA2D_ConfigOutputColor() API may be used instead if colors values formatting with respect to color mode is not done by the user code.

Reference Manual to LL API cross reference:

- OCOLR BLUE LL_DMA2D_SetOutputColor
- OCOLR GREEN LL_DMA2D_SetOutputColor
- OCOLR RED LL_DMA2D_SetOutputColor
- OCOLR ALPHA LL_DMA2D_SetOutputColor

LL_DMA2D_GetOutputColor

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetOutputColor (DMA2D_TypeDef * DMA2Dx)

Function description

Get DMA2D output color, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Output:** color value between Min_Data=0 and Max_Data=0xFFFFFFFF

Notes

- Alpha channel and red, green, blue color values must be retrieved from the returned value based on the output color mode (ARGB8888, RGB888, RGB565, ARGB1555 or ARGB4444) as set by LL_DMA2D_SetOutputColorMode.

Reference Manual to LL API cross reference:

- OCOLR BLUE LL_DMA2D_GetOutputColor
- OCOLR GREEN LL_DMA2D_GetOutputColor
- OCOLR RED LL_DMA2D_GetOutputColor
- OCOLR ALPHA LL_DMA2D_GetOutputColor

LL_DMA2D_SetLineWatermark

Function name

__STATIC_INLINE void LL_DMA2D_SetLineWatermark (DMA2D_TypeDef * DMA2Dx, uint32_t LineWatermark)

Function description

Set DMA2D line watermark, expressed on 16 bits ([15:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **LineWatermark:** Value between Min_Data=0 and Max_Data=0xFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- LWR LW LL_DMA2D_SetLineWatermark

LL_DMA2D_GetLineWatermark

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetLineWatermark (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D line watermark, expressed on 16 bits ([15:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Line:** watermark value between Min_Data=0 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- LWR LW LL_DMA2D_GetLineWatermark

LL_DMA2D_SetDeadTime

Function name

__STATIC_INLINE void LL_DMA2D_SetDeadTime (DMA2D_TypeDef * DMA2Dx, uint32_t DeadTime)

Function description

Set DMA2D dead time, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **DeadTime:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- AMTCR DT LL_DMA2D_SetDeadTime

LL_DMA2D_GetDeadTime

Function name

__STATIC_INLINE uint32_t LL_DMA2D_GetDeadTime (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D dead time, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Dead:** time value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- AMTCR DT LL_DMA2D_GetDeadTime

LL_DMA2D_EnableDeadTime

Function name

__STATIC_INLINE void LL_DMA2D_EnableDeadTime (DMA2D_TypeDef * DMA2Dx)

Function description

Enable DMA2D dead time functionality.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- AMTCR EN LL_DMA2D_EnableDeadTime

LL_DMA2D_DisableDeadTime

Function name

__STATIC_INLINE void LL_DMA2D_DisableDeadTime (DMA2D_TypeDef * DMA2Dx)

Function description

Disable DMA2D dead time functionality.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- AMTCR EN LL_DMA2D_DisableDeadTime

LL_DMA2D_IsEnabledDeadTime

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsEnabledDeadTime (DMA2D_TypeDef * DMA2Dx)

Function description

Indicate if DMA2D dead time functionality is enabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- AMTCR EN LL_DMA2D_IsEnabledDeadTime

LL_DMA2D_FGND_SetMemAddr

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetMemAddr (DMA2D_TypeDef * DMA2Dx, uint32_t MemoryAddress)

Function description

Set DMA2D foreground memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **MemoryAddress:** Value between Min_Data=0 and Max_Data=0xFFFFFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGMAR MA LL_DMA2D_FGND_SetMemAddr

LL_DMA2D_FGND_GetMemAddr

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetMemAddr (DMA2D_TypeDef * DMA2Dx)

Function description

Get DMA2D foreground memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Foreground:** memory address value between Min_Data=0 and Max_Data=0xFFFFFFFF

Reference Manual to LL API cross reference:

- FGMAR MA LL_DMA2D_FGND_GetMemAddr

LL_DMA2D_FGND_EnableCLUTLoad

Function name

__STATIC_INLINE void LL_DMA2D_FGND_EnableCLUTLoad (DMA2D_TypeDef * DMA2Dx)

Function description

Enable DMA2D foreground CLUT loading.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGPFCCR START LL_DMA2D_FGND_EnableCLUTLoad

LL_DMA2D_FGND_IsEnabledCLUTLoad

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_IsEnabledCLUTLoad (DMA2D_TypeDef * DMA2Dx)

Function description

Indicate if DMA2D foreground CLUT loading is enabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- FGPFCCR START LL_DMA2D_FGND_IsEnabledCLUTLoad

LL_DMA2D_FGND_SetColorMode

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetColorMode (DMA2D_TypeDef * DMA2Dx, uint32_t ColorMode)

Function description

Set DMA2D foreground color mode.

Parameters

- **DMA2Dx:** DMA2D Instance
- **ColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_INPUT_MODE_ARGB8888
  - LL_DMA2D_INPUT_MODE_RGB888
  - LL_DMA2D_INPUT_MODE_RGB565
  - LL_DMA2D_INPUT_MODE_ARGB1555
  - LL_DMA2D_INPUT_MODE_ARGB4444
  - LL_DMA2D_INPUT_MODE_L8
  - LL_DMA2D_INPUT_MODE_AL44
  - LL_DMA2D_INPUT_MODE_AL88
  - LL_DMA2D_INPUT_MODE_L4
  - LL_DMA2D_INPUT_MODE_A8
  - LL_DMA2D_INPUT_MODE_A4

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGPFCCR CM LL_DMA2D_FGND_SetColorMode

LL_DMA2D_FGND_GetColorMode

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetColorMode (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D foreground color mode.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA2D_INPUT_MODE_ARGB8888
  - LL_DMA2D_INPUT_MODE_RGB888
  - LL_DMA2D_INPUT_MODE_RGB565
  - LL_DMA2D_INPUT_MODE_ARGB1555
  - LL_DMA2D_INPUT_MODE_ARGB4444
  - LL_DMA2D_INPUT_MODE_L8
  - LL_DMA2D_INPUT_MODE_AL44
  - LL_DMA2D_INPUT_MODE_AL88
  - LL_DMA2D_INPUT_MODE_L4
  - LL_DMA2D_INPUT_MODE_A8
  - LL_DMA2D_INPUT_MODE_A4

Reference Manual to LL API cross reference:

- FGPFCCR CM LL_DMA2D_FGND_GetColorMode

LL_DMA2D_FGND_SetAlphaMode

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetAlphaMode (DMA2D_TypeDef * DMA2Dx, uint32_t AphaMode)

Function description

Set DMA2D foreground alpha mode.

Parameters

- **DMA2Dx:** DMA2D Instance
- **AphaMode:** This parameter can be one of the following values:
  - LL_DMA2D_ALPHA_MODE_NO_MODIF
  - LL_DMA2D_ALPHA_MODE_REPLACE
  - LL_DMA2D_ALPHA_MODE_COMBINE

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGPFCCR AM LL_DMA2D_FGND_SetAlphaMode

LL_DMA2D_FGND_GetAlphaMode

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetAlphaMode (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D foreground alpha mode.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA2D_ALPHA_MODE_NO_MODIF
  - LL_DMA2D_ALPHA_MODE_REPLACE
  - LL_DMA2D_ALPHA_MODE_COMBINE

Reference Manual to LL API cross reference:

- FGPFCCR AM LL_DMA2D_FGND_GetAlphaMode

LL_DMA2D_FGND_SetAlpha

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetAlpha (DMA2D_TypeDef * DMA2Dx, uint32_t Alpha)

Function description

Set DMA2D foreground alpha value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Alpha:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGPFCCR ALPHA LL_DMA2D_FGND_SetAlpha

LL_DMA2D_FGND_GetAlpha

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetAlpha (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D foreground alpha value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Alpha:** value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- FGPFCCR ALPHA LL_DMA2D_FGND_GetAlpha

LL_DMA2D_FGND_SetLineOffset

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetLineOffset (DMA2D_TypeDef * DMA2Dx, uint32_t LineOffset)

Function description

Set DMA2D foreground line offset, expressed on 14 bits ([13:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **LineOffset:** Value between Min_Data=0 and Max_Data=0x3FF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGOR LO LL_DMA2D_FGND_SetLineOffset

LL_DMA2D_FGND_GetLineOffset

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetLineOffset (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D foreground line offset, expressed on 14 bits ([13:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Foreground:** line offset value between Min_Data=0 and Max_Data=0x3FF

Reference Manual to LL API cross reference:

- FGOR LO LL_DMA2D_FGND_GetLineOffset

LL_DMA2D_FGND_SetColor

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetColor (DMA2D_TypeDef * DMA2Dx, uint32_t Red, uint32_t Green, uint32_t Blue)

Function description

Set DMA2D foreground color values, expressed on 24 bits ([23:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Red:** Value between Min_Data=0 and Max_Data=0xFF
- **Green:** Value between Min_Data=0 and Max_Data=0xFF
- **Blue:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGCOLR RED LL_DMA2D_FGND_SetColor
- FGCOLR GREEN LL_DMA2D_FGND_SetColor
- FGCOLR BLUE LL_DMA2D_FGND_SetColor

LL_DMA2D_FGND_SetRedColor

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetRedColor (DMA2D_TypeDef * DMA2Dx, uint32_t Red)

Function description

Set DMA2D foreground red color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Red:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGCOLR RED LL_DMA2D_FGND_SetRedColor

LL_DMA2D_FGND_GetRedColor

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetRedColor (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D foreground red color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Red:** color value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- FGCOLR RED LL_DMA2D_FGND_GetRedColor

LL_DMA2D_FGND_SetGreenColor

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetGreenColor (DMA2D_TypeDef * DMA2Dx, uint32_t Green)

Function description

Set DMA2D foreground green color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Green:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGCOLR GREEN LL_DMA2D_FGND_SetGreenColor

LL_DMA2D_FGND_GetGreenColor

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetGreenColor (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D foreground green color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Green:** color value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- FGCOLR GREEN LL_DMA2D_FGND_GetGreenColor

LL_DMA2D_FGND_SetBlueColor

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetBlueColor (DMA2D_TypeDef * DMA2Dx, uint32_t Blue)

Function description

Set DMA2D foreground blue color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Blue:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGCOLR BLUE LL_DMA2D_FGND_SetBlueColor

LL_DMA2D_FGND_GetBlueColor

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetBlueColor (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D foreground blue color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Blue:** color value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- FGCOLR BLUE LL_DMA2D_FGND_GetBlueColor

LL_DMA2D_FGND_SetCLUTMemAddr

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetCLUTMemAddr (DMA2D_TypeDef * DMA2Dx, uint32_t CLUTMemoryAddress)

Function description

Set DMA2D foreground CLUT memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **CLUTMemoryAddress:** Value between Min_Data=0 and Max_Data=0xFFFFFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGCMAR MA LL_DMA2D_FGND_SetCLUTMemAddr

LL_DMA2D_FGND_GetCLUTMemAddr

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetCLUTMemAddr (DMA2D_TypeDef * DMA2Dx)

Function description

Get DMA2D foreground CLUT memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Foreground:** CLUT memory address value between Min_Data=0 and Max_Data=0xFFFFFFFF

Reference Manual to LL API cross reference:

- FGCMAR MA LL_DMA2D_FGND_GetCLUTMemAddr

LL_DMA2D_FGND_SetCLUTSize

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetCLUTSize (DMA2D_TypeDef * DMA2Dx, uint32_t CLUTSize)

Function description

Set DMA2D foreground CLUT size, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **CLUTSize:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGPFCCR CS LL_DMA2D_FGND_SetCLUTSize

LL_DMA2D_FGND_GetCLUTSize

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetCLUTSize (DMA2D_TypeDef * DMA2Dx)

Function description

Get DMA2D foreground CLUT size, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Foreground:** CLUT size value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- FGPFCCR CS LL_DMA2D_FGND_GetCLUTSize

LL_DMA2D_FGND_SetCLUTColorMode

Function name

__STATIC_INLINE void LL_DMA2D_FGND_SetCLUTColorMode (DMA2D_TypeDef * DMA2Dx, uint32_t CLUTColorMode)

Function description

Set DMA2D foreground CLUT color mode.

Parameters

- **DMA2Dx:** DMA2D Instance
- **CLUTColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_CLUT_COLOR_MODE_ARGB8888
  - LL_DMA2D_CLUT_COLOR_MODE_RGB888

Return values

- **None:**

Reference Manual to LL API cross reference:

- FGPFCCR CCM LL_DMA2D_FGND_SetCLUTColorMode

LL_DMA2D_FGND_GetCLUTColorMode

Function name

__STATIC_INLINE uint32_t LL_DMA2D_FGND_GetCLUTColorMode (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D foreground CLUT color mode.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA2D_CLUT_COLOR_MODE_ARGB8888
  - LL_DMA2D_CLUT_COLOR_MODE_RGB888

Reference Manual to LL API cross reference:

- FGPFCCR CCM LL_DMA2D_FGND_GetCLUTColorMode

LL_DMA2D_BGND_SetMemAddr

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetMemAddr (DMA2D_TypeDef * DMA2Dx, uint32_t MemoryAddress)

Function description

Set DMA2D background memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **MemoryAddress:** Value between Min_Data=0 and Max_Data=0xFFFFFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGMAR MA LL_DMA2D_BGND_SetMemAddr

LL_DMA2D_BGND_GetMemAddr

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetMemAddr (DMA2D_TypeDef * DMA2Dx)

Function description

Get DMA2D background memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Background:** memory address value between Min_Data=0 and Max_Data=0xFFFFFFFF

Reference Manual to LL API cross reference:

- BGMAR MA LL_DMA2D_BGND_GetMemAddr

LL_DMA2D_BGND_EnableCLUTLoad

Function name

__STATIC_INLINE void LL_DMA2D_BGND_EnableCLUTLoad (DMA2D_TypeDef * DMA2Dx)

Function description

Enable DMA2D background CLUT loading.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGPFCCR START LL_DMA2D_BGND_EnableCLUTLoad

LL_DMA2D_BGND_IsEnabledCLUTLoad

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_IsEnabledCLUTLoad (DMA2D_TypeDef * DMA2Dx)

Function description

Indicate if DMA2D background CLUT loading is enabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- BGPFCCR START LL_DMA2D_BGND_IsEnabledCLUTLoad

LL_DMA2D_BGND_SetColorMode

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetColorMode (DMA2D_TypeDef * DMA2Dx, uint32_t ColorMode)

Function description

Set DMA2D background color mode.

Parameters

- **DMA2Dx:** DMA2D Instance
- **ColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_INPUT_MODE_ARGB8888
  - LL_DMA2D_INPUT_MODE_RGB888
  - LL_DMA2D_INPUT_MODE_RGB565
  - LL_DMA2D_INPUT_MODE_ARGB1555
  - LL_DMA2D_INPUT_MODE_ARGB4444
  - LL_DMA2D_INPUT_MODE_L8
  - LL_DMA2D_INPUT_MODE_AL44
  - LL_DMA2D_INPUT_MODE_AL88
  - LL_DMA2D_INPUT_MODE_L4
  - LL_DMA2D_INPUT_MODE_A8
  - LL_DMA2D_INPUT_MODE_A4

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGPFCCR CM LL_DMA2D_BGND_SetColorMode

LL_DMA2D_BGND_GetColorMode

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetColorMode (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D background color mode.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA2D_INPUT_MODE_ARGB8888
  - LL_DMA2D_INPUT_MODE_RGB888
  - LL_DMA2D_INPUT_MODE_RGB565
  - LL_DMA2D_INPUT_MODE_ARGB1555
  - LL_DMA2D_INPUT_MODE_ARGB4444
  - LL_DMA2D_INPUT_MODE_L8
  - LL_DMA2D_INPUT_MODE_AL44
  - LL_DMA2D_INPUT_MODE_AL88
  - LL_DMA2D_INPUT_MODE_L4
  - LL_DMA2D_INPUT_MODE_A8
  - LL_DMA2D_INPUT_MODE_A4

Reference Manual to LL API cross reference:

- BGPFCCR CM LL_DMA2D_BGND_GetColorMode

LL_DMA2D_BGND_SetAlphaMode

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetAlphaMode (DMA2D_TypeDef * DMA2Dx, uint32_t AphaMode)

Function description

Set DMA2D background alpha mode.

Parameters

- **DMA2Dx:** DMA2D Instance
- **AphaMode:** This parameter can be one of the following values:
  - LL_DMA2D_ALPHA_MODE_NO_MODIF
  - LL_DMA2D_ALPHA_MODE_REPLACE
  - LL_DMA2D_ALPHA_MODE_COMBINE

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGPFCCR AM LL_DMA2D_BGND_SetAlphaMode

LL_DMA2D_BGND_GetAlphaMode

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetAlphaMode (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D background alpha mode.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA2D_ALPHA_MODE_NO_MODIF
  - LL_DMA2D_ALPHA_MODE_REPLACE
  - LL_DMA2D_ALPHA_MODE_COMBINE

Reference Manual to LL API cross reference:

- BGPFCCR AM LL_DMA2D_BGND_GetAlphaMode

LL_DMA2D_BGND_SetAlpha

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetAlpha (DMA2D_TypeDef * DMA2Dx, uint32_t Alpha)

Function description

Set DMA2D background alpha value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Alpha:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGPFCCR ALPHA LL_DMA2D_BGND_SetAlpha

LL_DMA2D_BGND_GetAlpha

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetAlpha (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D background alpha value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Alpha:** value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- BGPFCCR ALPHA LL_DMA2D_BGND_GetAlpha

LL_DMA2D_BGND_SetLineOffset

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetLineOffset (DMA2D_TypeDef * DMA2Dx, uint32_t LineOffset)

Function description

Set DMA2D background line offset, expressed on 14 bits ([13:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **LineOffset:** Value between Min_Data=0 and Max_Data=0x3FF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGOR LO LL_DMA2D_BGND_SetLineOffset

LL_DMA2D_BGND_GetLineOffset

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetLineOffset (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D background line offset, expressed on 14 bits ([13:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Background:** line offset value between Min_Data=0 and Max_Data=0x3FF

Reference Manual to LL API cross reference:

- BGOR LO LL_DMA2D_BGND_GetLineOffset

LL_DMA2D_BGND_SetColor

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetColor (DMA2D_TypeDef * DMA2Dx, uint32_t Red, uint32_t Green, uint32_t Blue)

Function description

Set DMA2D background color values, expressed on 24 bits ([23:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Red:** Value between Min_Data=0 and Max_Data=0xFF
- **Green:** Value between Min_Data=0 and Max_Data=0xFF
- **Blue:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGCOLR RED LL_DMA2D_BGND_SetColor
- BGCOLR GREEN LL_DMA2D_BGND_SetColor
- BGCOLR BLUE LL_DMA2D_BGND_SetColor

LL_DMA2D_BGND_SetRedColor

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetRedColor (DMA2D_TypeDef * DMA2Dx, uint32_t Red)

Function description

Set DMA2D background red color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Red:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGCOLR RED LL_DMA2D_BGND_SetRedColor

LL_DMA2D_BGND_GetRedColor

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetRedColor (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D background red color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Red:** color value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- BGCOLR RED LL_DMA2D_BGND_GetRedColor

LL_DMA2D_BGND_SetGreenColor

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetGreenColor (DMA2D_TypeDef * DMA2Dx, uint32_t Green)

Function description

Set DMA2D background green color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Green:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGCOLR GREEN LL_DMA2D_BGND_SetGreenColor

LL_DMA2D_BGND_GetGreenColor

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetGreenColor (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D background green color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Green:** color value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- BGCOLR GREEN LL_DMA2D_BGND_GetGreenColor

LL_DMA2D_BGND_SetBlueColor

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetBlueColor (DMA2D_TypeDef * DMA2Dx, uint32_t Blue)

Function description

Set DMA2D background blue color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **Blue:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGCOLR BLUE LL_DMA2D_BGND_SetBlueColor

LL_DMA2D_BGND_GetBlueColor

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetBlueColor (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D background blue color value, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Blue:** color value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- BGCOLR BLUE LL_DMA2D_BGND_GetBlueColor

LL_DMA2D_BGND_SetCLUTMemAddr

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetCLUTMemAddr (DMA2D_TypeDef * DMA2Dx, uint32_t CLUTMemoryAddress)

Function description

Set DMA2D background CLUT memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **CLUTMemoryAddress:** Value between Min_Data=0 and Max_Data=0xFFFFFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGCMAR MA LL_DMA2D_BGND_SetCLUTMemAddr

LL_DMA2D_BGND_GetCLUTMemAddr

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetCLUTMemAddr (DMA2D_TypeDef * DMA2Dx)

Function description

Get DMA2D background CLUT memory address, expressed on 32 bits ([31:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Background:** CLUT memory address value between Min_Data=0 and Max_Data=0xFFFFFFFF

Reference Manual to LL API cross reference:

- BGCMAR MA LL_DMA2D_BGND_GetCLUTMemAddr

LL_DMA2D_BGND_SetCLUTSize

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetCLUTSize (DMA2D_TypeDef * DMA2Dx, uint32_t CLUTSize)

Function description

Set DMA2D background CLUT size, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance
- **CLUTSize:** Value between Min_Data=0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGPFCCR CS LL_DMA2D_BGND_SetCLUTSize

LL_DMA2D_BGND_GetCLUTSize

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetCLUTSize (DMA2D_TypeDef * DMA2Dx)

Function description

Get DMA2D background CLUT size, expressed on 8 bits ([7:0] bits).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Background:** CLUT size value between Min_Data=0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- BGPFCCR CS LL_DMA2D_BGND_GetCLUTSize

LL_DMA2D_BGND_SetCLUTColorMode

Function name

__STATIC_INLINE void LL_DMA2D_BGND_SetCLUTColorMode (DMA2D_TypeDef * DMA2Dx, uint32_t CLUTColorMode)

Function description

Set DMA2D background CLUT color mode.

Parameters

- **DMA2Dx:** DMA2D Instance
- **CLUTColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_CLUT_COLOR_MODE_ARGB8888
  - LL_DMA2D_CLUT_COLOR_MODE_RGB888

Return values

- **None:**

Reference Manual to LL API cross reference:

- BGPFCCR CCM LL_DMA2D_BGND_SetCLUTColorMode

LL_DMA2D_BGND_GetCLUTColorMode

Function name

__STATIC_INLINE uint32_t LL_DMA2D_BGND_GetCLUTColorMode (DMA2D_TypeDef * DMA2Dx)

Function description

Return DMA2D background CLUT color mode.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA2D_CLUT_COLOR_MODE_ARGB8888
  - LL_DMA2D_CLUT_COLOR_MODE_RGB888

Reference Manual to LL API cross reference:

- BGPFCCR CCM LL_DMA2D_BGND_GetCLUTColorMode

LL_DMA2D_IsActiveFlag_CE

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsActiveFlag_CE (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D Configuration Error Interrupt Flag is set or not.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR CEIF LL_DMA2D_IsActiveFlag_CE

LL_DMA2D_IsActiveFlag_CTC

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsActiveFlag_CTC (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D CLUT Transfer Complete Interrupt Flag is set or not.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR CTCIF LL_DMA2D_IsActiveFlag_CTC

LL_DMA2D_IsActiveFlag_CAE

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsActiveFlag_CAE (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D CLUT Access Error Interrupt Flag is set or not.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR CAEIF LL_DMA2D_IsActiveFlag_CAE

LL_DMA2D_IsActiveFlag_TW

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsActiveFlag_TW (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D Transfer Watermark Interrupt Flag is set or not.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR TWIF LL_DMA2D_IsActiveFlag_TW

LL_DMA2D_IsActiveFlag_TC

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsActiveFlag_TC (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D Transfer Complete Interrupt Flag is set or not.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR TCIF LL_DMA2D_IsActiveFlag_TC

LL_DMA2D_IsActiveFlag_TE

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsActiveFlag_TE (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D Transfer Error Interrupt Flag is set or not.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR TEIF LL_DMA2D_IsActiveFlag_TE

LL_DMA2D_ClearFlag_CE

Function name

__STATIC_INLINE void LL_DMA2D_ClearFlag_CE (DMA2D_TypeDef * DMA2Dx)

Function description

Clear DMA2D Configuration Error Interrupt Flag.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IFCR CCEIF LL_DMA2D_ClearFlag_CE

LL_DMA2D_ClearFlag_CTC

Function name

__STATIC_INLINE void LL_DMA2D_ClearFlag_CTC (DMA2D_TypeDef * DMA2Dx)

Function description

Clear DMA2D CLUT Transfer Complete Interrupt Flag.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IFCR CCTCIF LL_DMA2D_ClearFlag_CTC

LL_DMA2D_ClearFlag_CAE

Function name

__STATIC_INLINE void LL_DMA2D_ClearFlag_CAE (DMA2D_TypeDef * DMA2Dx)

Function description

Clear DMA2D CLUT Access Error Interrupt Flag.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IFCR CAECIF LL_DMA2D_ClearFlag_CAE

LL_DMA2D_ClearFlag_TW

Function name

__STATIC_INLINE void LL_DMA2D_ClearFlag_TW (DMA2D_TypeDef * DMA2Dx)

Function description

Clear DMA2D Transfer Watermark Interrupt Flag.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IFCR CTWIF LL_DMA2D_ClearFlag_TW

LL_DMA2D_ClearFlag_TC

Function name

__STATIC_INLINE void LL_DMA2D_ClearFlag_TC (DMA2D_TypeDef * DMA2Dx)

Function description

Clear DMA2D Transfer Complete Interrupt Flag.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IFCR CTCIF LL_DMA2D_ClearFlag_TC

LL_DMA2D_ClearFlag_TE

Function name

__STATIC_INLINE void LL_DMA2D_ClearFlag_TE (DMA2D_TypeDef * DMA2Dx)

Function description

Clear DMA2D Transfer Error Interrupt Flag.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IFCR CTEIF LL_DMA2D_ClearFlag_TE

LL_DMA2D_EnableIT_CE

Function name

__STATIC_INLINE void LL_DMA2D_EnableIT_CE (DMA2D_TypeDef * DMA2Dx)

Function description

Enable Configuration Error Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CEIE LL_DMA2D_EnableIT_CE

LL_DMA2D_EnableIT_CTC

Function name

__STATIC_INLINE void LL_DMA2D_EnableIT_CTC (DMA2D_TypeDef * DMA2Dx)

Function description

Enable CLUT Transfer Complete Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CTCIE LL_DMA2D_EnableIT_CTC

LL_DMA2D_EnableIT_CAE

Function name

__STATIC_INLINE void LL_DMA2D_EnableIT_CAE (DMA2D_TypeDef * DMA2Dx)

Function description

Enable CLUT Access Error Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CAEIE LL_DMA2D_EnableIT_CAE

LL_DMA2D_EnableIT_TW

Function name

__STATIC_INLINE void LL_DMA2D_EnableIT_TW (DMA2D_TypeDef * DMA2Dx)

Function description

Enable Transfer Watermark Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TWIE LL_DMA2D_EnableIT_TW

LL_DMA2D_EnableIT_TC

Function name

__STATIC_INLINE void LL_DMA2D_EnableIT_TC (DMA2D_TypeDef * DMA2Dx)

Function description

Enable Transfer Complete Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TCIE LL_DMA2D_EnableIT_TC

LL_DMA2D_EnableIT_TE

Function name

__STATIC_INLINE void LL_DMA2D_EnableIT_TE (DMA2D_TypeDef * DMA2Dx)

Function description

Enable Transfer Error Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TEIE LL_DMA2D_EnableIT_TE

LL_DMA2D_DisableIT_CE

Function name

__STATIC_INLINE void LL_DMA2D_DisableIT_CE (DMA2D_TypeDef * DMA2Dx)

Function description

Disable Configuration Error Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CEIE LL_DMA2D_DisableIT_CE

LL_DMA2D_DisableIT_CTC

Function name

__STATIC_INLINE void LL_DMA2D_DisableIT_CTC (DMA2D_TypeDef * DMA2Dx)

Function description

Disable CLUT Transfer Complete Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CTCIE LL_DMA2D_DisableIT_CTC

LL_DMA2D_DisableIT_CAE

Function name

__STATIC_INLINE void LL_DMA2D_DisableIT_CAE (DMA2D_TypeDef * DMA2Dx)

Function description

Disable CLUT Access Error Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CAEIE LL_DMA2D_DisableIT_CAE

LL_DMA2D_DisableIT_TW

Function name

__STATIC_INLINE void LL_DMA2D_DisableIT_TW (DMA2D_TypeDef * DMA2Dx)

Function description

Disable Transfer Watermark Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TWIE LL_DMA2D_DisableIT_TW

LL_DMA2D_DisableIT_TC

Function name

__STATIC_INLINE void LL_DMA2D_DisableIT_TC (DMA2D_TypeDef * DMA2Dx)

Function description

Disable Transfer Complete Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TCIE LL_DMA2D_DisableIT_TC

LL_DMA2D_DisableIT_TE

Function name

__STATIC_INLINE void LL_DMA2D_DisableIT_TE (DMA2D_TypeDef * DMA2Dx)

Function description

Disable Transfer Error Interrupt.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TEIE LL_DMA2D_DisableIT_TE

LL_DMA2D_IsEnabledIT_CE

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsEnabledIT_CE (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D Configuration Error interrupt source is enabled or disabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR CEIE LL_DMA2D_IsEnabledIT_CE

LL_DMA2D_IsEnabledIT_CTC

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsEnabledIT_CTC (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D CLUT Transfer Complete interrupt source is enabled or disabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR CTCIE LL_DMA2D_IsEnabledIT_CTC

LL_DMA2D_IsEnabledIT_CAE

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsEnabledIT_CAE (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D CLUT Access Error interrupt source is enabled or disabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR CAEIE LL_DMA2D_IsEnabledIT_CAE

LL_DMA2D_IsEnabledIT_TW

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsEnabledIT_TW (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D Transfer Watermark interrupt source is enabled or disabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR TWIE LL_DMA2D_IsEnabledIT_TW

LL_DMA2D_IsEnabledIT_TC

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsEnabledIT_TC (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D Transfer Complete interrupt source is enabled or disabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR TCIE LL_DMA2D_IsEnabledIT_TC

LL_DMA2D_IsEnabledIT_TE

Function name

__STATIC_INLINE uint32_t LL_DMA2D_IsEnabledIT_TE (DMA2D_TypeDef * DMA2Dx)

Function description

Check if the DMA2D Transfer Error interrupt source is enabled or disabled.

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR TEIE LL_DMA2D_IsEnabledIT_TE

LL_DMA2D_DeInit

Function name

ErrorStatus LL_DMA2D_DeInit (DMA2D_TypeDef * DMA2Dx)

Function description

De-initialize DMA2D registers (registers restored to their default values).

Parameters

- **DMA2Dx:** DMA2D Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: DMA2D registers are de-initialized
  - ERROR: DMA2D registers are not de-initialized

LL_DMA2D_Init

Function name

ErrorStatus LL_DMA2D_Init (DMA2D_TypeDef * DMA2Dx, LL_DMA2D_InitTypeDef * DMA2D_InitStruct)

Function description

Initialize DMA2D registers according to the specified parameters in DMA2D_InitStruct.

Parameters

- **DMA2Dx:** DMA2D Instance
- **DMA2D_InitStruct:** pointer to a LL_DMA2D_InitTypeDef structure that contains the configuration information for the specified DMA2D peripheral.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: DMA2D registers are initialized according to DMA2D_InitStruct content
  - ERROR: Issue occurred during DMA2D registers initialization

Notes

- DMA2D transfers must be disabled to set initialization bits in configuration registers, otherwise ERROR result is returned.

LL_DMA2D_StructInit

Function name

void LL_DMA2D_StructInit (LL_DMA2D_InitTypeDef * DMA2D_InitStruct)

Function description

Set each LL_DMA2D_InitTypeDef field to default value.

Parameters

- **DMA2D_InitStruct:** pointer to a LL_DMA2D_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

LL_DMA2D_ConfigLayer

Function name

void LL_DMA2D_ConfigLayer (DMA2D_TypeDef * DMA2Dx, LL_DMA2D_LayerCfgTypeDef * DMA2D_LayerCfg, uint32_t LayerIdx)

Function description

Configure the foreground or background according to the specified parameters in the LL_DMA2D_LayerCfgTypeDef structure.

Parameters

- **DMA2Dx:** DMA2D Instance
- **DMA2D_LayerCfg:** pointer to a LL_DMA2D_LayerCfgTypeDef structure that contains the configuration information for the specified layer.
- **LayerIdx:** DMA2D Layer index. This parameter can be one of the following values: 0(background) / 1(foreground)

Return values

- **None:**

LL_DMA2D_LayerCfgStructInit

Function name

void LL_DMA2D_LayerCfgStructInit (LL_DMA2D_LayerCfgTypeDef * DMA2D_LayerCfg)

Function description

Set each LL_DMA2D_LayerCfgTypeDef field to default value.

Parameters

- **DMA2D_LayerCfg:** pointer to a LL_DMA2D_LayerCfgTypeDef structure whose fields will be set to default values.

Return values

- **None:**

LL_DMA2D_ConfigOutputColor

Function name

void LL_DMA2D_ConfigOutputColor (DMA2D_TypeDef * DMA2Dx, LL_DMA2D_ColorTypeDef * DMA2D_ColorStruct)

Function description

Initialize DMA2D output color register according to the specified parameters in DMA2D_ColorStruct.

Parameters

- **DMA2Dx:** DMA2D Instance
- **DMA2D_ColorStruct:** pointer to a LL_DMA2D_ColorTypeDef structure that contains the color configuration information for the specified DMA2D peripheral.

Return values

- **None:**

LL_DMA2D_GetOutputBlueColor

Function name

uint32_t LL_DMA2D_GetOutputBlueColor (DMA2D_TypeDef * DMA2Dx, uint32_t ColorMode)

Function description

Return DMA2D output Blue color.

Parameters

- **DMA2Dx:** DMA2D Instance.
- **ColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_OUTPUT_MODE_ARGB8888
  - LL_DMA2D_OUTPUT_MODE_RGB888
  - LL_DMA2D_OUTPUT_MODE_RGB565
  - LL_DMA2D_OUTPUT_MODE_ARGB1555
  - LL_DMA2D_OUTPUT_MODE_ARGB4444

Return values

- **Output:** Blue color value between Min_Data=0 and Max_Data=0xFF

LL_DMA2D_GetOutputGreenColor

Function name

uint32_t LL_DMA2D_GetOutputGreenColor (DMA2D_TypeDef * DMA2Dx, uint32_t ColorMode)

Function description

Return DMA2D output Green color.

Parameters

- **DMA2Dx:** DMA2D Instance.
- **ColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_OUTPUT_MODE_ARGB8888
  - LL_DMA2D_OUTPUT_MODE_RGB888
  - LL_DMA2D_OUTPUT_MODE_RGB565
  - LL_DMA2D_OUTPUT_MODE_ARGB1555
  - LL_DMA2D_OUTPUT_MODE_ARGB4444

Return values

- **Output:** Green color value between Min_Data=0 and Max_Data=0xFF

LL_DMA2D_GetOutputRedColor

Function name

uint32_t LL_DMA2D_GetOutputRedColor (DMA2D_TypeDef * DMA2Dx, uint32_t ColorMode)

Function description

Return DMA2D output Red color.

Parameters

- **DMA2Dx:** DMA2D Instance.
- **ColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_OUTPUT_MODE_ARGB8888
  - LL_DMA2D_OUTPUT_MODE_RGB888
  - LL_DMA2D_OUTPUT_MODE_RGB565
  - LL_DMA2D_OUTPUT_MODE_ARGB1555
  - LL_DMA2D_OUTPUT_MODE_ARGB4444

Return values

- **Output:** Red color value between Min_Data=0 and Max_Data=0xFF

LL_DMA2D_GetOutputAlphaColor

Function name

uint32_t LL_DMA2D_GetOutputAlphaColor (DMA2D_TypeDef * DMA2Dx, uint32_t ColorMode)

Function description

Return DMA2D output Alpha color.

Parameters

- **DMA2Dx:** DMA2D Instance.
- **ColorMode:** This parameter can be one of the following values:
  - LL_DMA2D_OUTPUT_MODE_ARGB8888
  - LL_DMA2D_OUTPUT_MODE_RGB888
  - LL_DMA2D_OUTPUT_MODE_RGB565
  - LL_DMA2D_OUTPUT_MODE_ARGB1555
  - LL_DMA2D_OUTPUT_MODE_ARGB4444

Return values

- **Output:** Alpha color value between Min_Data=0 and Max_Data=0xFF

LL_DMA2D_ConfigSize

Function name

void LL_DMA2D_ConfigSize (DMA2D_TypeDef * DMA2Dx, uint32_t NbrOfLines, uint32_t NbrOfPixelsPerLines)

Function description

Configure DMA2D transfer size.

Parameters

- **DMA2Dx:** DMA2D Instance
- **NbrOfLines:** Value between Min_Data=0 and Max_Data=0xFFFF
- **NbrOfPixelsPerLines:** Value between Min_Data=0 and Max_Data=0x3FFF

Return values

- **None:**

78.3 DMA2D Firmware driver defines

The following section lists the various define and macros of the module.

78.3.1 DMA2D

DMA2D

Alpha Mode

LL_DMA2D_ALPHA_MODE_NO_MODIF

No modification of the alpha channel value

LL_DMA2D_ALPHA_MODE_REPLACE

Replace original alpha channel value by programmed alpha value

LL_DMA2D_ALPHA_MODE_COMBINE

Replace original alpha channel value by programmed alpha value with original alpha channel value

CLUT Color Mode

LL_DMA2D_CLUT_COLOR_MODE_ARGB8888

ARGB8888

LL_DMA2D_CLUT_COLOR_MODE_RGB888

RGB888

Get Flags Defines

LL_DMA2D_FLAG_CEIF

Configuration Error Interrupt Flag

LL_DMA2D_FLAG_CTCIF

CLUT Transfer Complete Interrupt Flag

LL_DMA2D_FLAG_CAEIF

CLUT Access Error Interrupt Flag

LL_DMA2D_FLAG_TWIF

Transfer Watermark Interrupt Flag

LL_DMA2D_FLAG_TCIF

Transfer Complete Interrupt Flag

LL_DMA2D_FLAG_TEIF

Transfer Error Interrupt Flag

Input Color Mode

LL_DMA2D_INPUT_MODE_ARGB8888

ARGB8888

LL_DMA2D_INPUT_MODE_RGB888

RGB888

LL_DMA2D_INPUT_MODE_RGB565

RGB565

LL_DMA2D_INPUT_MODE_ARGB1555

ARGB1555

LL_DMA2D_INPUT_MODE_ARGB4444

ARGB4444

LL_DMA2D_INPUT_MODE_L8

L8

LL_DMA2D_INPUT_MODE_AL44

AL44

LL_DMA2D_INPUT_MODE_AL88

AL88

LL_DMA2D_INPUT_MODE_L4

L4

LL_DMA2D_INPUT_MODE_A8

A8

LL_DMA2D_INPUT_MODE_A4

A4

IT Defines

LL_DMA2D_IT_CEIE

Configuration Error Interrupt

LL_DMA2D_IT_CTCIE

CLUT Transfer Complete Interrupt

LL_DMA2D_IT_CAEIE

CLUT Access Error Interrupt

LL_DMA2D_IT_TWIE

Transfer Watermark Interrupt

LL_DMA2D_IT_TCIE

Transfer Complete Interrupt

LL_DMA2D_IT_TEIE

Transfer Error Interrupt

Mode

LL_DMA2D_MODE_M2M

DMA2D memory to memory transfer mode

LL_DMA2D_MODE_M2M_PFC

DMA2D memory to memory with pixel format conversion transfer mode

LL_DMA2D_MODE_M2M_BLEND

DMA2D memory to memory with blending transfer mode

LL_DMA2D_MODE_R2M

DMA2D register to memory transfer mode

Output Color Mode

LL_DMA2D_OUTPUT_MODE_ARGB8888

ARGB8888

LL_DMA2D_OUTPUT_MODE_RGB888

RGB888

LL_DMA2D_OUTPUT_MODE_RGB565

RGB565

LL_DMA2D_OUTPUT_MODE_ARGB1555

ARGB1555

LL_DMA2D_OUTPUT_MODE_ARGB4444

ARGB4444

Common Write and read registers Macros

LL_DMA2D_WriteReg

Description:

- Write a value in DMA2D register.

Parameters:

- __INSTANCE__: DMA2D Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_DMA2D_ReadReg

Description:

- Read a value in DMA2D register.

Parameters:

- __INSTANCE__: DMA2D Instance
- __REG__: Register to be read

Return value:

- Register: value

79 LL DMA Generic Driver

79.1 DMA Firmware driver registers structures

79.1.1 LL_DMA_InitTypeDef

LL_DMA_InitTypeDef is defined in the stm32f4xx_ll_dma.h

Data Fields

- ***uint32_t PeriphOrM2MSrcAddress***
- ***uint32_t MemoryOrM2MDstAddress***
- ***uint32_t Direction***
- ***uint32_t Mode***
- ***uint32_t PeriphOrM2MSrcIncMode***
- ***uint32_t MemoryOrM2MDstIncMode***
- ***uint32_t PeriphOrM2MSrcDataSize***
- ***uint32_t MemoryOrM2MDstDataSize***
- ***uint32_t NbData***
- ***uint32_t Channel***
- ***uint32_t Priority***
- ***uint32_t FIFOMode***
- ***uint32_t FIFOThreshold***
- ***uint32_t MemBurst***
- ***uint32_t PeriphBurst***

Field Documentation

- ***uint32_t LL_DMA_InitTypeDef::PeriphOrM2MSrcAddress***

  Specifies the peripheral base address for DMA transfer or as Source base address in case of memory to memory transfer direction.This parameter must be a value between Min_Data = 0 and Max_Data = 0xFFFFFFFF.

- ***uint32_t LL_DMA_InitTypeDef::MemoryOrM2MDstAddress***

  Specifies the memory base address for DMA transfer or as Destination base address in case of memory to memory transfer direction.This parameter must be a value between Min_Data = 0 and Max_Data = 0xFFFFFFFF.

- ***uint32_t LL_DMA_InitTypeDef::Direction***

  Specifies if the data will be transferred from memory to peripheral, from memory to memory or from peripheral to memory. This parameter can be a value of ***DMA_LL_EC_DIRECTION***This feature can be modified afterwards using unitary function **LL_DMA_SetDataTransferDirection()**.

- ***uint32_t LL_DMA_InitTypeDef::Mode***

  Specifies the normal or circular operation mode. This parameter can be a value of ***DMA_LL_EC_MODE***

  Note:

  – The circular buffer mode cannot be used if the memory to memory data transfer direction is configured on the selected Stream

  This feature can be modified afterwards using unitary function **LL_DMA_SetMode()**.

- ***uint32_t LL_DMA_InitTypeDef::PeriphOrM2MSrcIncMode***

  Specifies whether the Peripheral address or Source address in case of memory to memory transfer direction is incremented or not. This parameter can be a value of ***DMA_LL_EC_PERIPH***This feature can be modified afterwards using unitary function **LL_DMA_SetPeriphIncMode()**.

- ***uint32_t LL_DMA_InitTypeDef::MemoryOrM2MDstIncMode***

  Specifies whether the Memory address or Destination address in case of memory to memory transfer direction is incremented or not. This parameter can be a value of ***DMA_LL_EC_MEMORY***This feature can be modified afterwards using unitary function **LL_DMA_SetMemoryIncMode()**.

- ***uint32_t LL_DMA_InitTypeDef::PeriphOrM2MSrcDataSize***

  Specifies the Peripheral data size alignment or Source data size alignment (byte, half word, word) in case of memory to memory transfer direction. This parameter can be a value of ***DMA_LL_EC_PDATAALIGN***This feature can be modified afterwards using unitary function **LL_DMA_SetPeriphSize()**.

- ***uint32_t LL_DMA_InitTypeDef::MemoryOrM2MDstDataSize***

  Specifies the Memory data size alignment or Destination data size alignment (byte, half word, word) in case of memory to memory transfer direction. This parameter can be a value of ***DMA_LL_EC_MDATAALIGN***This feature can be modified afterwards using unitary function **LL_DMA_SetMemorySize()**.

- ***uint32_t LL_DMA_InitTypeDef::NbData***

  Specifies the number of data to transfer, in data unit. The data unit is equal to the source buffer configuration set in PeripheralSize or MemorySize parameters depending in the transfer direction. This parameter must be a value between Min_Data = 0 and Max_Data = 0x0000FFFFThis feature can be modified afterwards using unitary function **LL_DMA_SetDataLength()**.

- ***uint32_t LL_DMA_InitTypeDef::Channel***

  Specifies the peripheral channel. This parameter can be a value of ***DMA_LL_EC_CHANNEL***This feature can be modified afterwards using unitary function **LL_DMA_SetChannelSelection()**.

- ***uint32_t LL_DMA_InitTypeDef::Priority***

  Specifies the channel priority level. This parameter can be a value of ***DMA_LL_EC_PRIORITY***This feature can be modified afterwards using unitary function **LL_DMA_SetStreamPriorityLevel()**.

- ***uint32_t LL_DMA_InitTypeDef::FIFOMode***

  Specifies if the FIFO mode or Direct mode will be used for the specified stream. This parameter can be a value of ***DMA_LL_FIFOMODE***

  Note:

  – The Direct mode (FIFO mode disabled) cannot be used if the memory-to-memory data transfer is configured on the selected stream

  This feature can be modified afterwards using unitary functions **LL_DMA_EnableFifoMode()** or **LL_DMA_EnableFifoMode()** .

- ***uint32_t LL_DMA_InitTypeDef::FIFOThreshold***

  Specifies the FIFO threshold level. This parameter can be a value of ***DMA_LL_EC_FIFOTHRESHOLD***This feature can be modified afterwards using unitary function **LL_DMA_SetFIFOThreshold()**.

- ***uint32_t LL_DMA_InitTypeDef::MemBurst***

  Specifies the Burst transfer configuration for the memory transfers. It specifies the amount of data to be transferred in a single non interruptible transaction. This parameter can be a value of ***DMA_LL_EC_MBURST***

  Note:

  – The burst mode is possible only if the address Increment mode is enabled.

  This feature can be modified afterwards using unitary function **LL_DMA_SetMemoryBurstxfer()**.

- ***uint32_t LL_DMA_InitTypeDef::PeriphBurst***

  Specifies the Burst transfer configuration for the peripheral transfers. It specifies the amount of data to be transferred in a single non interruptible transaction. This parameter can be a value of ***DMA_LL_EC_PBURST***

  Note:

  – The burst mode is possible only if the address Increment mode is enabled.

  This feature can be modified afterwards using unitary function **LL_DMA_SetPeriphBurstxfer()**.

79.2 DMA Firmware driver API description

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

79.2.1 Detailed description of functions

LL_DMA_EnableStream

Function name

__STATIC_INLINE void LL_DMA_EnableStream (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Enable DMA stream.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR EN LL_DMA_EnableStream

LL_DMA_DisableStream

Function name

__STATIC_INLINE void LL_DMA_DisableStream (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Disable DMA stream.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR EN LL_DMA_DisableStream

LL_DMA_IsEnabledStream

Function name

__STATIC_INLINE uint32_t LL_DMA_IsEnabledStream (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Check if DMA stream is enabled or disabled.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR EN LL_DMA_IsEnabledStream

LL_DMA_ConfigTransfer

Function name

__STATIC_INLINE void LL_DMA_ConfigTransfer (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Configuration)

Function description

Configure all parameters linked to DMA transfer.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Configuration:** This parameter must be a combination of all the following values:
  - LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH or LL_DMA_DIRECTION_MEMORY_TO_MEMORY
  - LL_DMA_MODE_NORMAL or LL_DMA_MODE_CIRCULAR or LL_DMA_MODE_PFCTRL
  - LL_DMA_PERIPH_INCREMENT or LL_DMA_PERIPH_NOINCREMENT
  - LL_DMA_MEMORY_INCREMENT or LL_DMA_MEMORY_NOINCREMENT
  - LL_DMA_PDATAALIGN_BYTE or LL_DMA_PDATAALIGN_HALFWORD or LL_DMA_PDATAALIGN_WORD
  - LL_DMA_MDATAALIGN_BYTE or LL_DMA_MDATAALIGN_HALFWORD or LL_DMA_MDATAALIGN_WORD
  - LL_DMA_PRIORITY_LOW or LL_DMA_PRIORITY_MEDIUM or LL_DMA_PRIORITY_HIGH or LL_DMA_PRIORITY_VERYHIGH

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DIR LL_DMA_ConfigTransfer
- CR CIRC LL_DMA_ConfigTransfer
- CR PINC LL_DMA_ConfigTransfer
- CR MINC LL_DMA_ConfigTransfer
- CR PSIZE LL_DMA_ConfigTransfer
- CR MSIZE LL_DMA_ConfigTransfer
- CR PL LL_DMA_ConfigTransfer
- CR PFCTRL LL_DMA_ConfigTransfer

LL_DMA_SetDataTransferDirection

Function name

__STATIC_INLINE void LL_DMA_SetDataTransferDirection (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Direction)

Function description

Set Data transfer direction (read from peripheral or from memory).

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Direction:** This parameter can be one of the following values:
  - LL_DMA_DIRECTION_PERIPH_TO_MEMORY
  - LL_DMA_DIRECTION_MEMORY_TO_PERIPH
  - LL_DMA_DIRECTION_MEMORY_TO_MEMORY

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DIR LL_DMA_SetDataTransferDirection

LL_DMA_GetDataTransferDirection

Function name

__STATIC_INLINE uint32_t LL_DMA_GetDataTransferDirection (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Data transfer direction (read from peripheral or from memory).

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_DIRECTION_PERIPH_TO_MEMORY
  - LL_DMA_DIRECTION_MEMORY_TO_PERIPH
  - LL_DMA_DIRECTION_MEMORY_TO_MEMORY

Reference Manual to LL API cross reference:

- CR DIR LL_DMA_GetDataTransferDirection

LL_DMA_SetMode

Function name

__STATIC_INLINE void LL_DMA_SetMode (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Mode)

Function description

Set DMA mode normal, circular or peripheral flow control.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Mode:** This parameter can be one of the following values:
  - LL_DMA_MODE_NORMAL
  - LL_DMA_MODE_CIRCULAR
  - LL_DMA_MODE_PFCTRL

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CIRC LL_DMA_SetMode
- CR PFCTRL LL_DMA_SetMode

LL_DMA_GetMode

Function name

__STATIC_INLINE uint32_t LL_DMA_GetMode (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get DMA mode normal, circular or peripheral flow control.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_MODE_NORMAL
  - LL_DMA_MODE_CIRCULAR
  - LL_DMA_MODE_PFCTRL

Reference Manual to LL API cross reference:

- CR CIRC LL_DMA_GetMode
- CR PFCTRL LL_DMA_GetMode

LL_DMA_SetPeriphIncMode

Function name

__STATIC_INLINE void LL_DMA_SetPeriphIncMode (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t IncrementMode)

Function description

Set Peripheral increment mode.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **IncrementMode:** This parameter can be one of the following values:
  - LL_DMA_PERIPH_NOINCREMENT
  - LL_DMA_PERIPH_INCREMENT

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PINC LL_DMA_SetPeriphIncMode

LL_DMA_GetPeriphIncMode

Function name

__STATIC_INLINE uint32_t LL_DMA_GetPeriphIncMode (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Peripheral increment mode.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_PERIPH_NOINCREMENT
  - LL_DMA_PERIPH_INCREMENT

Reference Manual to LL API cross reference:

- CR PINC LL_DMA_GetPeriphIncMode

LL_DMA_SetMemoryIncMode

Function name

__STATIC_INLINE void LL_DMA_SetMemoryIncMode (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t IncrementMode)

Function description

Set Memory increment mode.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **IncrementMode:** This parameter can be one of the following values:
  - LL_DMA_MEMORY_NOINCREMENT
  - LL_DMA_MEMORY_INCREMENT

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR MINC LL_DMA_SetMemoryIncMode

LL_DMA_GetMemoryIncMode

Function name

__STATIC_INLINE uint32_t LL_DMA_GetMemoryIncMode (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Memory increment mode.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_MEMORY_NOINCREMENT
  - LL_DMA_MEMORY_INCREMENT

Reference Manual to LL API cross reference:

- CR MINC LL_DMA_GetMemoryIncMode

LL_DMA_SetPeriphSize

Function name

__STATIC_INLINE void LL_DMA_SetPeriphSize (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Size)

Function description

Set Peripheral size.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Size:** This parameter can be one of the following values:
  - LL_DMA_PDATAALIGN_BYTE
  - LL_DMA_PDATAALIGN_HALFWORD
  - LL_DMA_PDATAALIGN_WORD

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PSIZE LL_DMA_SetPeriphSize

LL_DMA_GetPeriphSize

Function name

__STATIC_INLINE uint32_t LL_DMA_GetPeriphSize (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Peripheral size.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_PDATAALIGN_BYTE
  - LL_DMA_PDATAALIGN_HALFWORD
  - LL_DMA_PDATAALIGN_WORD

Reference Manual to LL API cross reference:

- CR PSIZE LL_DMA_GetPeriphSize

LL_DMA_SetMemorySize

Function name

__STATIC_INLINE void LL_DMA_SetMemorySize (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Size)

Function description

Set Memory size.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Size:** This parameter can be one of the following values:
  - LL_DMA_MDATAALIGN_BYTE
  - LL_DMA_MDATAALIGN_HALFWORD
  - LL_DMA_MDATAALIGN_WORD

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR MSIZE LL_DMA_SetMemorySize

LL_DMA_GetMemorySize

Function name

__STATIC_INLINE uint32_t LL_DMA_GetMemorySize (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Memory size.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_MDATAALIGN_BYTE
  - LL_DMA_MDATAALIGN_HALFWORD
  - LL_DMA_MDATAALIGN_WORD

Reference Manual to LL API cross reference:

- CR MSIZE LL_DMA_GetMemorySize

LL_DMA_SetIncOffsetSize

Function name

__STATIC_INLINE void LL_DMA_SetIncOffsetSize (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t OffsetSize)

Function description

Set Peripheral increment offset size.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **OffsetSize:** This parameter can be one of the following values:
  - LL_DMA_OFFSETSIZE_PSIZE
  - LL_DMA_OFFSETSIZE_FIXEDTO4

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PINCOS LL_DMA_SetIncOffsetSize

LL_DMA_GetIncOffsetSize

Function name

__STATIC_INLINE uint32_t LL_DMA_GetIncOffsetSize (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Peripheral increment offset size.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_OFFSETSIZE_PSIZE
  - LL_DMA_OFFSETSIZE_FIXEDTO4

Reference Manual to LL API cross reference:

- CR PINCOS LL_DMA_GetIncOffsetSize

LL_DMA_SetStreamPriorityLevel

Function name

__STATIC_INLINE void LL_DMA_SetStreamPriorityLevel (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Priority)

Function description

Set Stream priority level.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Priority:** This parameter can be one of the following values:
  - LL_DMA_PRIORITY_LOW
  - LL_DMA_PRIORITY_MEDIUM
  - LL_DMA_PRIORITY_HIGH
  - LL_DMA_PRIORITY_VERYHIGH

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PL LL_DMA_SetStreamPriorityLevel

LL_DMA_GetStreamPriorityLevel

Function name

__STATIC_INLINE uint32_t LL_DMA_GetStreamPriorityLevel (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Stream priority level.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_PRIORITY_LOW
  - LL_DMA_PRIORITY_MEDIUM
  - LL_DMA_PRIORITY_HIGH
  - LL_DMA_PRIORITY_VERYHIGH

Reference Manual to LL API cross reference:

- CR PL LL_DMA_GetStreamPriorityLevel

LL_DMA_SetDataLength

Function name

__STATIC_INLINE void LL_DMA_SetDataLength (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t NbData)

Function description

Set Number of data to transfer.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **NbData:** Between 0 to 0xFFFFFFFF

Return values

- **None:**

Notes

- This action has no effect if stream is enabled.

Reference Manual to LL API cross reference:

- NDTR NDT LL_DMA_SetDataLength

LL_DMA_GetDataLength

Function name

__STATIC_INLINE uint32_t LL_DMA_GetDataLength (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Number of data to transfer.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Between:** 0 to 0xFFFFFFFF

Notes

- Once the stream is enabled, the return value indicate the remaining bytes to be transmitted.

Reference Manual to LL API cross reference:

- NDTR NDT LL_DMA_GetDataLength

LL_DMA_SetChannelSelection

Function name

__STATIC_INLINE void LL_DMA_SetChannelSelection (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Channel)

Function description

Select Channel number associated to the Stream.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Channel:** This parameter can be one of the following values:
  - LL_DMA_CHANNEL_0
  - LL_DMA_CHANNEL_1
  - LL_DMA_CHANNEL_2
  - LL_DMA_CHANNEL_3
  - LL_DMA_CHANNEL_4
  - LL_DMA_CHANNEL_5
  - LL_DMA_CHANNEL_6
  - LL_DMA_CHANNEL_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CHSEL LL_DMA_SetChannelSelection

LL_DMA_GetChannelSelection

Function name

__STATIC_INLINE uint32_t LL_DMA_GetChannelSelection (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get the Channel number associated to the Stream.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_CHANNEL_0
  - LL_DMA_CHANNEL_1
  - LL_DMA_CHANNEL_2
  - LL_DMA_CHANNEL_3
  - LL_DMA_CHANNEL_4
  - LL_DMA_CHANNEL_5
  - LL_DMA_CHANNEL_6
  - LL_DMA_CHANNEL_7

Reference Manual to LL API cross reference:

- CR CHSEL LL_DMA_GetChannelSelection

LL_DMA_SetMemoryBurstxfer

Function name

__STATIC_INLINE void LL_DMA_SetMemoryBurstxfer (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Mburst)

Function description

Set Memory burst transfer configuration.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Mburst:** This parameter can be one of the following values:
  - LL_DMA_MBURST_SINGLE
  - LL_DMA_MBURST_INC4
  - LL_DMA_MBURST_INC8
  - LL_DMA_MBURST_INC16

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR MBURST LL_DMA_SetMemoryBurstxfer

LL_DMA_GetMemoryBurstxfer

Function name

__STATIC_INLINE uint32_t LL_DMA_GetMemoryBurstxfer (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Memory burst transfer configuration.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_MBURST_SINGLE
  - LL_DMA_MBURST_INC4
  - LL_DMA_MBURST_INC8
  - LL_DMA_MBURST_INC16

Reference Manual to LL API cross reference:

- CR MBURST LL_DMA_GetMemoryBurstxfer

LL_DMA_SetPeriphBurstxfer

Function name

__STATIC_INLINE void LL_DMA_SetPeriphBurstxfer (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Pburst)

Function description

Set Peripheral burst transfer configuration.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Pburst:** This parameter can be one of the following values:
  - LL_DMA_PBURST_SINGLE
  - LL_DMA_PBURST_INC4
  - LL_DMA_PBURST_INC8
  - LL_DMA_PBURST_INC16

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PBURST LL_DMA_SetPeriphBurstxfer

LL_DMA_GetPeriphBurstxfer

Function name

__STATIC_INLINE uint32_t LL_DMA_GetPeriphBurstxfer (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Peripheral burst transfer configuration.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_PBURST_SINGLE
  - LL_DMA_PBURST_INC4
  - LL_DMA_PBURST_INC8
  - LL_DMA_PBURST_INC16

Reference Manual to LL API cross reference:

- CR PBURST LL_DMA_GetPeriphBurstxfer

LL_DMA_SetCurrentTargetMem

Function name

__STATIC_INLINE void LL_DMA_SetCurrentTargetMem (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t CurrentMemory)

Function description

Set Current target (only in double buffer mode) to Memory 1 or Memory 0.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **CurrentMemory:** This parameter can be one of the following values:
  - LL_DMA_CURRENTTARGETMEM0
  - LL_DMA_CURRENTTARGETMEM1

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CT LL_DMA_SetCurrentTargetMem

LL_DMA_GetCurrentTargetMem

Function name

__STATIC_INLINE uint32_t LL_DMA_GetCurrentTargetMem (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Set Current target (only in double buffer mode) to Memory 1 or Memory 0.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_CURRENTTARGETMEM0
  - LL_DMA_CURRENTTARGETMEM1

Reference Manual to LL API cross reference:

- CR CT LL_DMA_GetCurrentTargetMem

LL_DMA_EnableDoubleBufferMode

Function name

__STATIC_INLINE void LL_DMA_EnableDoubleBufferMode (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Enable the double buffer mode.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DBM LL_DMA_EnableDoubleBufferMode

LL_DMA_DisableDoubleBufferMode

Function name

__STATIC_INLINE void LL_DMA_DisableDoubleBufferMode (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Disable the double buffer mode.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DBM LL_DMA_DisableDoubleBufferMode

LL_DMA_GetFIFOStatus

Function name

__STATIC_INLINE uint32_t LL_DMA_GetFIFOStatus (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get FIFO status.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_FIFOSTATUS_0_25
  - LL_DMA_FIFOSTATUS_25_50
  - LL_DMA_FIFOSTATUS_50_75
  - LL_DMA_FIFOSTATUS_75_100
  - LL_DMA_FIFOSTATUS_EMPTY
  - LL_DMA_FIFOSTATUS_FULL

Reference Manual to LL API cross reference:

- FCR FS LL_DMA_GetFIFOStatus

LL_DMA_DisableFifoMode

Function name

__STATIC_INLINE void LL_DMA_DisableFifoMode (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Disable Fifo mode.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- FCR DMDIS LL_DMA_DisableFifoMode

LL_DMA_EnableFifoMode

Function name

__STATIC_INLINE void LL_DMA_EnableFifoMode (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Enable Fifo mode.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- FCR DMDIS LL_DMA_EnableFifoMode

LL_DMA_SetFIFOThreshold

Function name

__STATIC_INLINE void LL_DMA_SetFIFOThreshold (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Threshold)

Function description

Select FIFO threshold.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Threshold:** This parameter can be one of the following values:
  - LL_DMA_FIFOTHRESHOLD_1_4
  - LL_DMA_FIFOTHRESHOLD_1_2
  - LL_DMA_FIFOTHRESHOLD_3_4
  - LL_DMA_FIFOTHRESHOLD_FULL

Return values

- **None:**

Reference Manual to LL API cross reference:

- FCR FTH LL_DMA_SetFIFOThreshold

LL_DMA_GetFIFOThreshold

Function name

__STATIC_INLINE uint32_t LL_DMA_GetFIFOThreshold (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get FIFO threshold.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Returned:** value can be one of the following values:
  - LL_DMA_FIFOTHRESHOLD_1_4
  - LL_DMA_FIFOTHRESHOLD_1_2
  - LL_DMA_FIFOTHRESHOLD_3_4
  - LL_DMA_FIFOTHRESHOLD_FULL

Reference Manual to LL API cross reference:

- FCR FTH LL_DMA_GetFIFOThreshold

LL_DMA_ConfigFifo

Function name

__STATIC_INLINE void LL_DMA_ConfigFifo (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t FifoMode, uint32_t FifoThreshold)

Function description

Configure the FIFO .

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **FifoMode:** This parameter can be one of the following values:
  - LL_DMA_FIFOMODE_ENABLE
  - LL_DMA_FIFOMODE_DISABLE
- **FifoThreshold:** This parameter can be one of the following values:
  - LL_DMA_FIFOTHRESHOLD_1_4
  - LL_DMA_FIFOTHRESHOLD_1_2
  - LL_DMA_FIFOTHRESHOLD_3_4
  - LL_DMA_FIFOTHRESHOLD_FULL

Return values

- **None:**

Reference Manual to LL API cross reference:

- FCR FTH LL_DMA_ConfigFifo
- FCR DMDIS LL_DMA_ConfigFifo

LL_DMA_ConfigAddresses

Function name

__STATIC_INLINE void LL_DMA_ConfigAddresses (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t SrcAddress, uint32_t DstAddress, uint32_t Direction)

Function description

Configure the Source and Destination addresses.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **SrcAddress:** Between 0 to 0xFFFFFFFF
- **DstAddress:** Between 0 to 0xFFFFFFFF
- **Direction:** This parameter can be one of the following values:
  - LL_DMA_DIRECTION_PERIPH_TO_MEMORY
  - LL_DMA_DIRECTION_MEMORY_TO_PERIPH
  - LL_DMA_DIRECTION_MEMORY_TO_MEMORY

Return values

- **None:**

Notes

- This API must not be called when the DMA stream is enabled.

Reference Manual to LL API cross reference:

- M0AR M0A LL_DMA_ConfigAddresses
- PAR PA LL_DMA_ConfigAddresses

LL_DMA_SetMemoryAddress

Function name

__STATIC_INLINE void LL_DMA_SetMemoryAddress (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t MemoryAddress)

Function description

Set the Memory address.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **MemoryAddress:** Between 0 to 0xFFFFFFFF

Return values

- **None:**

Notes

- Interface used for direction LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH only.
- This API must not be called when the DMA channel is enabled.

Reference Manual to LL API cross reference:

- M0AR M0A LL_DMA_SetMemoryAddress

LL_DMA_SetPeriphAddress

Function name

__STATIC_INLINE void LL_DMA_SetPeriphAddress (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t PeriphAddress)

Function description

Set the Peripheral address.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **PeriphAddress:** Between 0 to 0xFFFFFFFF

Return values

- **None:**

Notes

- Interface used for direction LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH only.
- This API must not be called when the DMA channel is enabled.

Reference Manual to LL API cross reference:

- PAR PA LL_DMA_SetPeriphAddress

LL_DMA_GetMemoryAddress

Function name

__STATIC_INLINE uint32_t LL_DMA_GetMemoryAddress (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get the Memory address.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Between:** 0 to 0xFFFFFFFF

Notes

- Interface used for direction LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH only.

Reference Manual to LL API cross reference:

- M0AR M0A LL_DMA_GetMemoryAddress

LL_DMA_GetPeriphAddress

Function name

__STATIC_INLINE uint32_t LL_DMA_GetPeriphAddress (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get the Peripheral address.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Between:** 0 to 0xFFFFFFFF

Notes

- Interface used for direction LL_DMA_DIRECTION_PERIPH_TO_MEMORY or LL_DMA_DIRECTION_MEMORY_TO_PERIPH only.

Reference Manual to LL API cross reference:

- PAR PA LL_DMA_GetPeriphAddress

LL_DMA_SetM2MSrcAddress

Function name

__STATIC_INLINE void LL_DMA_SetM2MSrcAddress (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t MemoryAddress)

Function description

Set the Memory to Memory Source address.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **MemoryAddress:** Between 0 to 0xFFFFFFFF

Return values

- **None:**

Notes

- Interface used for direction LL_DMA_DIRECTION_MEMORY_TO_MEMORY only.
- This API must not be called when the DMA channel is enabled.

Reference Manual to LL API cross reference:

- PAR PA LL_DMA_SetM2MSrcAddress

LL_DMA_SetM2MDstAddress

Function name

__STATIC_INLINE void LL_DMA_SetM2MDstAddress (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t MemoryAddress)

Function description

Set the Memory to Memory Destination address.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **MemoryAddress:** Between 0 to 0xFFFFFFFF

Return values

- **None:**

Notes

- Interface used for direction LL_DMA_DIRECTION_MEMORY_TO_MEMORY only.
- This API must not be called when the DMA channel is enabled.

Reference Manual to LL API cross reference:

- M0AR M0A LL_DMA_SetM2MDstAddress

LL_DMA_GetM2MSrcAddress

Function name

__STATIC_INLINE uint32_t LL_DMA_GetM2MSrcAddress (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get the Memory to Memory Source address.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Between:** 0 to 0xFFFFFFFF

Notes

- Interface used for direction LL_DMA_DIRECTION_MEMORY_TO_MEMORY only.

Reference Manual to LL API cross reference:

- PAR PA LL_DMA_GetM2MSrcAddress

LL_DMA_GetM2MDstAddress

Function name

__STATIC_INLINE uint32_t LL_DMA_GetM2MDstAddress (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get the Memory to Memory Destination address.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Between:** 0 to 0xFFFFFFFF

Notes

- Interface used for direction LL_DMA_DIRECTION_MEMORY_TO_MEMORY only.

Reference Manual to LL API cross reference:

- M0AR M0A LL_DMA_GetM2MDstAddress

LL_DMA_SetMemory1Address

Function name

__STATIC_INLINE void LL_DMA_SetMemory1Address (DMA_TypeDef * DMAx, uint32_t Stream, uint32_t Address)

Function description

Set Memory 1 address (used in case of Double buffer mode).

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **Address:** Between 0 to 0xFFFFFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- M1AR M1A LL_DMA_SetMemory1Address

LL_DMA_GetMemory1Address

Function name

__STATIC_INLINE uint32_t LL_DMA_GetMemory1Address (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Get Memory 1 address (used in case of Double buffer mode).

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **Between:** 0 to 0xFFFFFFFF

Reference Manual to LL API cross reference:

- M1AR M1A LL_DMA_GetMemory1Address

LL_DMA_IsActiveFlag_HT0

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT0 (DMA_TypeDef * DMAx)

Function description

Get Stream 0 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR HTIF0 LL_DMA_IsActiveFlag_HT0

LL_DMA_IsActiveFlag_HT1

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT1 (DMA_TypeDef * DMAx)

Function description

Get Stream 1 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR HTIF1 LL_DMA_IsActiveFlag_HT1

LL_DMA_IsActiveFlag_HT2

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT2 (DMA_TypeDef * DMAx)

Function description

Get Stream 2 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR HTIF2 LL_DMA_IsActiveFlag_HT2

LL_DMA_IsActiveFlag_HT3

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT3 (DMA_TypeDef * DMAx)

Function description

Get Stream 3 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR HTIF3 LL_DMA_IsActiveFlag_HT3

LL_DMA_IsActiveFlag_HT4

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT4 (DMA_TypeDef * DMAx)

Function description

Get Stream 4 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR HTIF4 LL_DMA_IsActiveFlag_HT4

LL_DMA_IsActiveFlag_HT5

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT5 (DMA_TypeDef * DMAx)

Function description

Get Stream 5 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR HTIF0 LL_DMA_IsActiveFlag_HT5

LL_DMA_IsActiveFlag_HT6

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT6 (DMA_TypeDef * DMAx)

Function description

Get Stream 6 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR HTIF6 LL_DMA_IsActiveFlag_HT6

LL_DMA_IsActiveFlag_HT7

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_HT7 (DMA_TypeDef * DMAx)

Function description

Get Stream 7 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR HTIF7 LL_DMA_IsActiveFlag_HT7

LL_DMA_IsActiveFlag_TC0

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC0 (DMA_TypeDef * DMAx)

Function description

Get Stream 0 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR TCIF0 LL_DMA_IsActiveFlag_TC0

LL_DMA_IsActiveFlag_TC1

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC1 (DMA_TypeDef * DMAx)

Function description

Get Stream 1 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR TCIF1 LL_DMA_IsActiveFlag_TC1

LL_DMA_IsActiveFlag_TC2

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC2 (DMA_TypeDef * DMAx)

Function description

Get Stream 2 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR TCIF2 LL_DMA_IsActiveFlag_TC2

LL_DMA_IsActiveFlag_TC3

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC3 (DMA_TypeDef * DMAx)

Function description

Get Stream 3 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR TCIF3 LL_DMA_IsActiveFlag_TC3

LL_DMA_IsActiveFlag_TC4

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC4 (DMA_TypeDef * DMAx)

Function description

Get Stream 4 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR TCIF4 LL_DMA_IsActiveFlag_TC4

LL_DMA_IsActiveFlag_TC5

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC5 (DMA_TypeDef * DMAx)

Function description

Get Stream 5 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR TCIF0 LL_DMA_IsActiveFlag_TC5

LL_DMA_IsActiveFlag_TC6

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC6 (DMA_TypeDef * DMAx)

Function description

Get Stream 6 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR TCIF6 LL_DMA_IsActiveFlag_TC6

LL_DMA_IsActiveFlag_TC7

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TC7 (DMA_TypeDef * DMAx)

Function description

Get Stream 7 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR TCIF7 LL_DMA_IsActiveFlag_TC7

LL_DMA_IsActiveFlag_TE0

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE0 (DMA_TypeDef * DMAx)

Function description

Get Stream 0 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR TEIF0 LL_DMA_IsActiveFlag_TE0

LL_DMA_IsActiveFlag_TE1

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE1 (DMA_TypeDef * DMAx)

Function description

Get Stream 1 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR TEIF1 LL_DMA_IsActiveFlag_TE1

LL_DMA_IsActiveFlag_TE2

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE2 (DMA_TypeDef * DMAx)

Function description

Get Stream 2 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR TEIF2 LL_DMA_IsActiveFlag_TE2

LL_DMA_IsActiveFlag_TE3

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE3 (DMA_TypeDef * DMAx)

Function description

Get Stream 3 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR TEIF3 LL_DMA_IsActiveFlag_TE3

LL_DMA_IsActiveFlag_TE4

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE4 (DMA_TypeDef * DMAx)

Function description

Get Stream 4 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR TEIF4 LL_DMA_IsActiveFlag_TE4

LL_DMA_IsActiveFlag_TE5

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE5 (DMA_TypeDef * DMAx)

Function description

Get Stream 5 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR TEIF0 LL_DMA_IsActiveFlag_TE5

LL_DMA_IsActiveFlag_TE6

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE6 (DMA_TypeDef * DMAx)

Function description

Get Stream 6 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR TEIF6 LL_DMA_IsActiveFlag_TE6

LL_DMA_IsActiveFlag_TE7

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_TE7 (DMA_TypeDef * DMAx)

Function description

Get Stream 7 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR TEIF7 LL_DMA_IsActiveFlag_TE7

LL_DMA_IsActiveFlag_DME0

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_DME0 (DMA_TypeDef * DMAx)

Function description

Get Stream 0 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR DMEIF0 LL_DMA_IsActiveFlag_DME0

LL_DMA_IsActiveFlag_DME1

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_DME1 (DMA_TypeDef * DMAx)

Function description

Get Stream 1 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR DMEIF1 LL_DMA_IsActiveFlag_DME1

LL_DMA_IsActiveFlag_DME2

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_DME2 (DMA_TypeDef * DMAx)

Function description

Get Stream 2 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR DMEIF2 LL_DMA_IsActiveFlag_DME2

LL_DMA_IsActiveFlag_DME3

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_DME3 (DMA_TypeDef * DMAx)

Function description

Get Stream 3 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR DMEIF3 LL_DMA_IsActiveFlag_DME3

LL_DMA_IsActiveFlag_DME4

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_DME4 (DMA_TypeDef * DMAx)

Function description

Get Stream 4 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR DMEIF4 LL_DMA_IsActiveFlag_DME4

LL_DMA_IsActiveFlag_DME5

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_DME5 (DMA_TypeDef * DMAx)

Function description

Get Stream 5 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR DMEIF0 LL_DMA_IsActiveFlag_DME5

LL_DMA_IsActiveFlag_DME6

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_DME6 (DMA_TypeDef * DMAx)

Function description

Get Stream 6 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR DMEIF6 LL_DMA_IsActiveFlag_DME6

LL_DMA_IsActiveFlag_DME7

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_DME7 (DMA_TypeDef * DMAx)

Function description

Get Stream 7 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR DMEIF7 LL_DMA_IsActiveFlag_DME7

LL_DMA_IsActiveFlag_FE0

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_FE0 (DMA_TypeDef * DMAx)

Function description

Get Stream 0 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR FEIF0 LL_DMA_IsActiveFlag_FE0

LL_DMA_IsActiveFlag_FE1

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_FE1 (DMA_TypeDef * DMAx)

Function description

Get Stream 1 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR FEIF1 LL_DMA_IsActiveFlag_FE1

LL_DMA_IsActiveFlag_FE2

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_FE2 (DMA_TypeDef * DMAx)

Function description

Get Stream 2 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR FEIF2 LL_DMA_IsActiveFlag_FE2

LL_DMA_IsActiveFlag_FE3

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_FE3 (DMA_TypeDef * DMAx)

Function description

Get Stream 3 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LISR FEIF3 LL_DMA_IsActiveFlag_FE3

LL_DMA_IsActiveFlag_FE4

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_FE4 (DMA_TypeDef * DMAx)

Function description

Get Stream 4 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR FEIF4 LL_DMA_IsActiveFlag_FE4

LL_DMA_IsActiveFlag_FE5

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_FE5 (DMA_TypeDef * DMAx)

Function description

Get Stream 5 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR FEIF0 LL_DMA_IsActiveFlag_FE5

LL_DMA_IsActiveFlag_FE6

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_FE6 (DMA_TypeDef * DMAx)

Function description

Get Stream 6 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR FEIF6 LL_DMA_IsActiveFlag_FE6

LL_DMA_IsActiveFlag_FE7

Function name

__STATIC_INLINE uint32_t LL_DMA_IsActiveFlag_FE7 (DMA_TypeDef * DMAx)

Function description

Get Stream 7 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- HISR FEIF7 LL_DMA_IsActiveFlag_FE7

LL_DMA_ClearFlag_HT0

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_HT0 (DMA_TypeDef * DMAx)

Function description

Clear Stream 0 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CHTIF0 LL_DMA_ClearFlag_HT0

LL_DMA_ClearFlag_HT1

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_HT1 (DMA_TypeDef * DMAx)

Function description

Clear Stream 1 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CHTIF1 LL_DMA_ClearFlag_HT1

LL_DMA_ClearFlag_HT2

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_HT2 (DMA_TypeDef * DMAx)

Function description

Clear Stream 2 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CHTIF2 LL_DMA_ClearFlag_HT2

LL_DMA_ClearFlag_HT3

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_HT3 (DMA_TypeDef * DMAx)

Function description

Clear Stream 3 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CHTIF3 LL_DMA_ClearFlag_HT3

LL_DMA_ClearFlag_HT4

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_HT4 (DMA_TypeDef * DMAx)

Function description

Clear Stream 4 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CHTIF4 LL_DMA_ClearFlag_HT4

LL_DMA_ClearFlag_HT5

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_HT5 (DMA_TypeDef * DMAx)

Function description

Clear Stream 5 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CHTIF5 LL_DMA_ClearFlag_HT5

LL_DMA_ClearFlag_HT6

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_HT6 (DMA_TypeDef * DMAx)

Function description

Clear Stream 6 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CHTIF6 LL_DMA_ClearFlag_HT6

LL_DMA_ClearFlag_HT7

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_HT7 (DMA_TypeDef * DMAx)

Function description

Clear Stream 7 half transfer flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CHTIF7 LL_DMA_ClearFlag_HT7

LL_DMA_ClearFlag_TC0

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TC0 (DMA_TypeDef * DMAx)

Function description

Clear Stream 0 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CTCIF0 LL_DMA_ClearFlag_TC0

LL_DMA_ClearFlag_TC1

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TC1 (DMA_TypeDef * DMAx)

Function description

Clear Stream 1 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CTCIF1 LL_DMA_ClearFlag_TC1

LL_DMA_ClearFlag_TC2

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TC2 (DMA_TypeDef * DMAx)

Function description

Clear Stream 2 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CTCIF2 LL_DMA_ClearFlag_TC2

LL_DMA_ClearFlag_TC3

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TC3 (DMA_TypeDef * DMAx)

Function description

Clear Stream 3 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CTCIF3 LL_DMA_ClearFlag_TC3

LL_DMA_ClearFlag_TC4

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TC4 (DMA_TypeDef * DMAx)

Function description

Clear Stream 4 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CTCIF4 LL_DMA_ClearFlag_TC4

LL_DMA_ClearFlag_TC5

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TC5 (DMA_TypeDef * DMAx)

Function description

Clear Stream 5 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CTCIF5 LL_DMA_ClearFlag_TC5

LL_DMA_ClearFlag_TC6

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TC6 (DMA_TypeDef * DMAx)

Function description

Clear Stream 6 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CTCIF6 LL_DMA_ClearFlag_TC6

LL_DMA_ClearFlag_TC7

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TC7 (DMA_TypeDef * DMAx)

Function description

Clear Stream 7 transfer complete flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CTCIF7 LL_DMA_ClearFlag_TC7

LL_DMA_ClearFlag_TE0

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TE0 (DMA_TypeDef * DMAx)

Function description

Clear Stream 0 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CTEIF0 LL_DMA_ClearFlag_TE0

LL_DMA_ClearFlag_TE1

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TE1 (DMA_TypeDef * DMAx)

Function description

Clear Stream 1 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CTEIF1 LL_DMA_ClearFlag_TE1

LL_DMA_ClearFlag_TE2

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TE2 (DMA_TypeDef * DMAx)

Function description

Clear Stream 2 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CTEIF2 LL_DMA_ClearFlag_TE2

LL_DMA_ClearFlag_TE3

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TE3 (DMA_TypeDef * DMAx)

Function description

Clear Stream 3 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CTEIF3 LL_DMA_ClearFlag_TE3

LL_DMA_ClearFlag_TE4

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TE4 (DMA_TypeDef * DMAx)

Function description

Clear Stream 4 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CTEIF4 LL_DMA_ClearFlag_TE4

LL_DMA_ClearFlag_TE5

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TE5 (DMA_TypeDef * DMAx)

Function description

Clear Stream 5 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CTEIF5 LL_DMA_ClearFlag_TE5

LL_DMA_ClearFlag_TE6

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TE6 (DMA_TypeDef * DMAx)

Function description

Clear Stream 6 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CTEIF6 LL_DMA_ClearFlag_TE6

LL_DMA_ClearFlag_TE7

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_TE7 (DMA_TypeDef * DMAx)

Function description

Clear Stream 7 transfer error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CTEIF7 LL_DMA_ClearFlag_TE7

LL_DMA_ClearFlag_DME0

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_DME0 (DMA_TypeDef * DMAx)

Function description

Clear Stream 0 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CDMEIF0 LL_DMA_ClearFlag_DME0

LL_DMA_ClearFlag_DME1

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_DME1 (DMA_TypeDef * DMAx)

Function description

Clear Stream 1 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CDMEIF1 LL_DMA_ClearFlag_DME1

LL_DMA_ClearFlag_DME2

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_DME2 (DMA_TypeDef * DMAx)

Function description

Clear Stream 2 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CDMEIF2 LL_DMA_ClearFlag_DME2

LL_DMA_ClearFlag_DME3

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_DME3 (DMA_TypeDef * DMAx)

Function description

Clear Stream 3 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CDMEIF3 LL_DMA_ClearFlag_DME3

LL_DMA_ClearFlag_DME4

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_DME4 (DMA_TypeDef * DMAx)

Function description

Clear Stream 4 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CDMEIF4 LL_DMA_ClearFlag_DME4

LL_DMA_ClearFlag_DME5

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_DME5 (DMA_TypeDef * DMAx)

Function description

Clear Stream 5 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CDMEIF5 LL_DMA_ClearFlag_DME5

LL_DMA_ClearFlag_DME6

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_DME6 (DMA_TypeDef * DMAx)

Function description

Clear Stream 6 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CDMEIF6 LL_DMA_ClearFlag_DME6

LL_DMA_ClearFlag_DME7

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_DME7 (DMA_TypeDef * DMAx)

Function description

Clear Stream 7 direct mode error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CDMEIF7 LL_DMA_ClearFlag_DME7

LL_DMA_ClearFlag_FE0

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_FE0 (DMA_TypeDef * DMAx)

Function description

Clear Stream 0 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CFEIF0 LL_DMA_ClearFlag_FE0

LL_DMA_ClearFlag_FE1

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_FE1 (DMA_TypeDef * DMAx)

Function description

Clear Stream 1 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CFEIF1 LL_DMA_ClearFlag_FE1

LL_DMA_ClearFlag_FE2

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_FE2 (DMA_TypeDef * DMAx)

Function description

Clear Stream 2 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CFEIF2 LL_DMA_ClearFlag_FE2

LL_DMA_ClearFlag_FE3

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_FE3 (DMA_TypeDef * DMAx)

Function description

Clear Stream 3 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- LIFCR CFEIF3 LL_DMA_ClearFlag_FE3

LL_DMA_ClearFlag_FE4

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_FE4 (DMA_TypeDef * DMAx)

Function description

Clear Stream 4 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CFEIF4 LL_DMA_ClearFlag_FE4

LL_DMA_ClearFlag_FE5

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_FE5 (DMA_TypeDef * DMAx)

Function description

Clear Stream 5 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CFEIF5 LL_DMA_ClearFlag_FE5

LL_DMA_ClearFlag_FE6

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_FE6 (DMA_TypeDef * DMAx)

Function description

Clear Stream 6 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CFEIF6 LL_DMA_ClearFlag_FE6

LL_DMA_ClearFlag_FE7

Function name

__STATIC_INLINE void LL_DMA_ClearFlag_FE7 (DMA_TypeDef * DMAx)

Function description

Clear Stream 7 FIFO error flag.

Parameters

- **DMAx:** DMAx Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- HIFCR CFEIF7 LL_DMA_ClearFlag_FE7

LL_DMA_EnableIT_HT

Function name

__STATIC_INLINE void LL_DMA_EnableIT_HT (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Enable Half transfer interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR HTIE LL_DMA_EnableIT_HT

LL_DMA_EnableIT_TE

Function name

__STATIC_INLINE void LL_DMA_EnableIT_TE (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Enable Transfer error interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TEIE LL_DMA_EnableIT_TE

LL_DMA_EnableIT_TC

Function name

__STATIC_INLINE void LL_DMA_EnableIT_TC (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Enable Transfer complete interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TCIE LL_DMA_EnableIT_TC

LL_DMA_EnableIT_DME

Function name

__STATIC_INLINE void LL_DMA_EnableIT_DME (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Enable Direct mode error interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DMEIE LL_DMA_EnableIT_DME

LL_DMA_EnableIT_FE

Function name

__STATIC_INLINE void LL_DMA_EnableIT_FE (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Enable FIFO error interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- FCR FEIE LL_DMA_EnableIT_FE

LL_DMA_DisableIT_HT

Function name

__STATIC_INLINE void LL_DMA_DisableIT_HT (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Disable Half transfer interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR HTIE LL_DMA_DisableIT_HT

LL_DMA_DisableIT_TE

Function name

__STATIC_INLINE void LL_DMA_DisableIT_TE (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Disable Transfer error interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TEIE LL_DMA_DisableIT_TE

LL_DMA_DisableIT_TC

Function name

__STATIC_INLINE void LL_DMA_DisableIT_TC (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Disable Transfer complete interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR TCIE LL_DMA_DisableIT_TC

LL_DMA_DisableIT_DME

Function name

__STATIC_INLINE void LL_DMA_DisableIT_DME (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Disable Direct mode error interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DMEIE LL_DMA_DisableIT_DME

LL_DMA_DisableIT_FE

Function name

__STATIC_INLINE void LL_DMA_DisableIT_FE (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Disable FIFO error interrupt.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- FCR FEIE LL_DMA_DisableIT_FE

LL_DMA_IsEnabledIT_HT

Function name

__STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_HT (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Check if Half transfer interrup is enabled.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR HTIE LL_DMA_IsEnabledIT_HT

LL_DMA_IsEnabledIT_TE

Function name

__STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_TE (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Check if Transfer error nterrup is enabled.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR TEIE LL_DMA_IsEnabledIT_TE

LL_DMA_IsEnabledIT_TC

Function name

__STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_TC (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Check if Transfer complete interrup is enabled.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR TCIE LL_DMA_IsEnabledIT_TC

LL_DMA_IsEnabledIT_DME

Function name

__STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_DME (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Check if Direct mode error interrupt is enabled.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR DMEIE LL_DMA_IsEnabledIT_DME

LL_DMA_IsEnabledIT_FE

Function name

__STATIC_INLINE uint32_t LL_DMA_IsEnabledIT_FE (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

Check if FIFO error interrup is enabled.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- FCR FEIE LL_DMA_IsEnabledIT_FE

LL_DMA_Init

Function name

uint32_t LL_DMA_Init (DMA_TypeDef * DMAx, uint32_t Stream, LL_DMA_InitTypeDef * DMA_InitStruct)

Function description

Initialize the DMA registers according to the specified parameters in DMA_InitStruct.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
- **DMA_InitStruct:** pointer to a LL_DMA_InitTypeDef structure.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: DMA registers are initialized
  - ERROR: Not applicable

Notes

- To convert DMAx_Streamy Instance to DMAx Instance and Streamy, use helper macros : __LL_DMA_GET_INSTANCE __LL_DMA_GET_STREAM

LL_DMA_DeInit

Function name

uint32_t LL_DMA_DeInit (DMA_TypeDef * DMAx, uint32_t Stream)

Function description

De-initialize the DMA registers to their default reset values.

Parameters

- **DMAx:** DMAx Instance
- **Stream:** This parameter can be one of the following values:
  - LL_DMA_STREAM_0
  - LL_DMA_STREAM_1
  - LL_DMA_STREAM_2
  - LL_DMA_STREAM_3
  - LL_DMA_STREAM_4
  - LL_DMA_STREAM_5
  - LL_DMA_STREAM_6
  - LL_DMA_STREAM_7
  - LL_DMA_STREAM_ALL

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: DMA registers are de-initialized
  - ERROR: DMA registers are not de-initialized

LL_DMA_StructInit

Function name

void LL_DMA_StructInit (LL_DMA_InitTypeDef * DMA_InitStruct)

Function description

Set each LL_DMA_InitTypeDef field to default value.

Parameters

- **DMA_InitStruct:** Pointer to a LL_DMA_InitTypeDef structure.

Return values

- **None:**

79.3 DMA Firmware driver defines

The following section lists the various define and macros of the module.

79.3.1 DMA

DMA

CHANNEL

LL_DMA_CHANNEL_0

LL_DMA_CHANNEL_1

LL_DMA_CHANNEL_2

LL_DMA_CHANNEL_3

LL_DMA_CHANNEL_4

LL_DMA_CHANNEL_5

LL_DMA_CHANNEL_6

LL_DMA_CHANNEL_7

CURRENTTARGETMEM

LL_DMA_CURRENTTARGETMEM0

Set CurrentTarget Memory to Memory 0

LL_DMA_CURRENTTARGETMEM1

Set CurrentTarget Memory to Memory 1

DIRECTION

LL_DMA_DIRECTION_PERIPH_TO_MEMORY

Peripheral to memory direction

LL_DMA_DIRECTION_MEMORY_TO_PERIPH

Memory to peripheral direction

LL_DMA_DIRECTION_MEMORY_TO_MEMORY

Memory to memory direction

DOUBLEBUFFER MODE

LL_DMA_DOUBLEBUFFER_MODE_DISABLE

Disable double buffering mode

LL_DMA_DOUBLEBUFFER_MODE_ENABLE

Enable double buffering mode

FIFOSTATUS 0

LL_DMA_FIFOSTATUS_0_25

0 < fifo_level < 1/4

LL_DMA_FIFOSTATUS_25_50

1/4 < fifo_level < 1/2

LL_DMA_FIFOSTATUS_50_75

1/2 < fifo_level < 3/4

LL_DMA_FIFOSTATUS_75_100

3/4 < fifo_level < full

LL_DMA_FIFOSTATUS_EMPTY

FIFO is empty

LL_DMA_FIFOSTATUS_FULL

FIFO is full

FIFOTHRESHOLD

LL_DMA_FIFOTHRESHOLD_1_4

FIFO threshold 1 quart full configuration

LL_DMA_FIFOTHRESHOLD_1_2

FIFO threshold half full configuration

LL_DMA_FIFOTHRESHOLD_3_4

FIFO threshold 3 quarts full configuration

LL_DMA_FIFOTHRESHOLD_FULL

FIFO threshold full configuration

MBURST

LL_DMA_MBURST_SINGLE

Memory burst single transfer configuration

LL_DMA_MBURST_INC4

Memory burst of 4 beats transfer configuration

LL_DMA_MBURST_INC8

Memory burst of 8 beats transfer configuration

LL_DMA_MBURST_INC16

Memory burst of 16 beats transfer configuration

MDATAALIGN

LL_DMA_MDATAALIGN_BYTE

Memory data alignment : Byte

LL_DMA_MDATAALIGN_HALFWORD

Memory data alignment : HalfWord

LL_DMA_MDATAALIGN_WORD

Memory data alignment : Word

MEMORY

LL_DMA_MEMORY_NOINCREMENT

Memory increment mode Disable

LL_DMA_MEMORY_INCREMENT

Memory increment mode Enable

MODE

LL_DMA_MODE_NORMAL

Normal Mode

LL_DMA_MODE_CIRCULAR

Circular Mode

LL_DMA_MODE_PFCTRL

Peripheral flow control mode

OFFSETSIZE

LL_DMA_OFFSETSIZE_PSIZE

Peripheral increment offset size is linked to the PSIZE

LL_DMA_OFFSETSIZE_FIXEDTO4

Peripheral increment offset size is fixed to 4 (32-bit alignment)

PBURST

LL_DMA_PBURST_SINGLE

Peripheral burst single transfer configuration

LL_DMA_PBURST_INC4

Peripheral burst of 4 beats transfer configuration

LL_DMA_PBURST_INC8

Peripheral burst of 8 beats transfer configuration

LL_DMA_PBURST_INC16

Peripheral burst of 16 beats transfer configuration

PDATAALIGN

LL_DMA_PDATAALIGN_BYTE

Peripheral data alignment : Byte

LL_DMA_PDATAALIGN_HALFWORD

Peripheral data alignment : HalfWord

LL_DMA_PDATAALIGN_WORD

Peripheral data alignment : Word

PERIPH

LL_DMA_PERIPH_NOINCREMENT

Peripheral increment mode Disable

LL_DMA_PERIPH_INCREMENT

Peripheral increment mode Enable

PRIORITY

LL_DMA_PRIORITY_LOW

Priority level : Low

LL_DMA_PRIORITY_MEDIUM

Priority level : Medium

LL_DMA_PRIORITY_HIGH

Priority level : High

LL_DMA_PRIORITY_VERYHIGH

Priority level : Very_High

STREAM

LL_DMA_STREAM_0

LL_DMA_STREAM_1

LL_DMA_STREAM_2

LL_DMA_STREAM_3

LL_DMA_STREAM_4

LL_DMA_STREAM_5

LL_DMA_STREAM_6

LL_DMA_STREAM_7

LL_DMA_STREAM_ALL

Convert DMAxStreamy

__LL_DMA_GET_INSTANCE

Description:

- Convert DMAx_Streamy into DMAx.

Parameters:

- __STREAM_INSTANCE__: DMAx_Streamy

Return value:

- DMAx

__LL_DMA_GET_STREAM

Description:

- Convert DMAx_Streamy into LL_DMA_STREAM_y.

Parameters:

- __STREAM_INSTANCE__: DMAx_Streamy

Return value:

- LL_DMA_CHANNEL_y

__LL_DMA_GET_STREAM_INSTANCE

Description:

- Convert DMA Instance DMAx and LL_DMA_STREAM_y into DMAx_Streamy.

Parameters:

- __DMA_INSTANCE__: DMAx
- __STREAM__: LL_DMA_STREAM_y

Return value:

- DMAx_Streamy

Common Write and read registers macros

LL_DMA_WriteReg

Description:

- Write a value in DMA register.

Parameters:

- __INSTANCE__: DMA Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_DMA_ReadReg

Description:

- Read a value in DMA register.

Parameters:

- __INSTANCE__: DMA Instance
- __REG__: Register to be read

Return value:

- Register: value

DMA_LL_FIFOMODE

LL_DMA_FIFOMODE_DISABLE

FIFO mode disable (direct mode is enabled)

LL_DMA_FIFOMODE_ENABLE

FIFO mode enable

80 LL FMPI2C Generic Driver

80.1 FMPI2C Firmware driver registers structures

80.1.1 LL_FMPI2C_InitTypeDef

LL_FMPI2C_InitTypeDef is defined in the stm32f4xx_ll_fmpi2c.h

Data Fields

- ***uint32_t PeripheralMode***
- ***uint32_t Timing***
- ***uint32_t AnalogFilter***
- ***uint32_t DigitalFilter***
- ***uint32_t OwnAddress1***
- ***uint32_t TypeAcknowledge***
- ***uint32_t OwnAddrSize***

Field Documentation

- ***uint32_t LL_FMPI2C_InitTypeDef::PeripheralMode***

  Specifies the peripheral mode. This parameter can be a value of ***FMPI2C_LL_EC_PERIPHERAL_MODE***This feature can be modified afterwards using unitary function **LL_FMPI2C_SetMode()**.
- ***uint32_t LL_FMPI2C_InitTypeDef::Timing***

  Specifies the SDA setup, hold time and the SCL high, low period values. This parameter must be set by referring to the STM32CubeMX Tool and the helper macro **__LL_FMPI2C_CONVERT_TIMINGS()**This feature can be modified afterwards using unitary function **LL_FMPI2C_SetTiming()**.
- ***uint32_t LL_FMPI2C_InitTypeDef::AnalogFilter***

  Enables or disables analog noise filter. This parameter can be a value of ***FMPI2C_LL_EC_ANALOGFILTER_SELECTION***This feature can be modified afterwards using unitary functions **LL_FMPI2C_EnableAnalogFilter()** or **LL_FMPI2C_DisableAnalogFilter()**.
- ***uint32_t LL_FMPI2C_InitTypeDef::DigitalFilter***

  Configures the digital noise filter. This parameter can be a number between Min_Data = 0x00 and Max_Data = 0x0FThis feature can be modified afterwards using unitary function **LL_FMPI2C_SetDigitalFilter()**.
- ***uint32_t LL_FMPI2C_InitTypeDef::OwnAddress1***

  Specifies the device own address 1. This parameter must be a value between Min_Data = 0x00 and Max_Data = 0x3FFThis feature can be modified afterwards using unitary function **LL_FMPI2C_SetOwnAddress1()**.
- ***uint32_t LL_FMPI2C_InitTypeDef::TypeAcknowledge***

  Specifies the ACKnowledge or Non ACKnowledge condition after the address receive match code or next received byte. This parameter can be a value of ***FMPI2C_LL_EC_I2C_ACKNOWLEDGE***This feature can be modified afterwards using unitary function **LL_FMPI2C_AcknowledgeNextData()**.
- ***uint32_t LL_FMPI2C_InitTypeDef::OwnAddrSize***

  Specifies the device own address 1 size (7-bit or 10-bit). This parameter can be a value of ***FMPI2C_LL_EC_OWNADDRESS1***This feature can be modified afterwards using unitary function **LL_FMPI2C_SetOwnAddress1()**.

80.2 FMPI2C Firmware driver API description

The following section lists the various functions of the FMPI2C library.

80.2.1 Detailed description of functions

LL_FMPI2C_Enable

Function name

__STATIC_INLINE void LL_FMPI2C_Enable (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable FMPI2C peripheral (PE = 1).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 PE LL_FMPI2C_Enable

LL_FMPI2C_Disable

Function name

__STATIC_INLINE void LL_FMPI2C_Disable (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable FMPI2C peripheral (PE = 0).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- When PE = 0, the FMPI2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.

Reference Manual to LL API cross reference:

- CR1 PE LL_FMPI2C_Disable

LL_FMPI2C_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabled (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if the FMPI2C peripheral is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 PE LL_FMPI2C_IsEnabled

LL_FMPI2C_ConfigFilters

Function name

__STATIC_INLINE void LL_FMPI2C_ConfigFilters (FMPI2C_TypeDef * FMPI2Cx, uint32_t AnalogFilter, uint32_t DigitalFilter)

Function description

Configure Noise Filters (Analog and Digital).

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **AnalogFilter:** This parameter can be one of the following values:
  - LL_FMPI2C_ANALOGFILTER_ENABLE
  - LL_FMPI2C_ANALOGFILTER_DISABLE
- **DigitalFilter:** This parameter must be a value between Min_Data=0x00 (Digital filter disabled) and Max_Data=0x0F (Digital filter enabled and filtering capability up to 15*tfmpi2cclk). This parameter is used to configure the digital noise filter on SDA and SCL input. The digital filter will filter spikes with a length of up to DNF[3:0]*tfmpi2cclk.

Return values

- **None:**

Notes

- If the analog filter is also enabled, the digital filter is added to analog filter. The filters can only be programmed when the FMPI2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR1 ANFOFF LL_FMPI2C_ConfigFilters
- CR1 DNF LL_FMPI2C_ConfigFilters

LL_FMPI2C_SetDigitalFilter

Function name

__STATIC_INLINE void LL_FMPI2C_SetDigitalFilter (FMPI2C_TypeDef * FMPI2Cx, uint32_t DigitalFilter)

Function description

Configure Digital Noise Filter.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **DigitalFilter:** This parameter must be a value between Min_Data=0x00 (Digital filter disabled) and Max_Data=0x0F (Digital filter enabled and filtering capability up to 15*tfmpi2cclk). This parameter is used to configure the digital noise filter on SDA and SCL input. The digital filter will filter spikes with a length of up to DNF[3:0]*tfmpi2cclk.

Return values

- **None:**

Notes

- If the analog filter is also enabled, the digital filter is added to analog filter. This filter can only be programmed when the FMPI2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR1 DNF LL_FMPI2C_SetDigitalFilter

LL_FMPI2C_GetDigitalFilter

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetDigitalFilter (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the current Digital Noise Filter configuration.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x0 and Max_Data=0xF

Reference Manual to LL API cross reference:

- CR1 DNF LL_FMPI2C_GetDigitalFilter

LL_FMPI2C_EnableAnalogFilter

Function name

__STATIC_INLINE void LL_FMPI2C_EnableAnalogFilter (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable Analog Noise Filter.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- This filter can only be programmed when the FMPI2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR1 ANFOFF LL_FMPI2C_EnableAnalogFilter

LL_FMPI2C_DisableAnalogFilter

Function name

__STATIC_INLINE void LL_FMPI2C_DisableAnalogFilter (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable Analog Noise Filter.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- This filter can only be programmed when the FMPI2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR1 ANFOFF LL_FMPI2C_DisableAnalogFilter

LL_FMPI2C_IsEnabledAnalogFilter

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledAnalogFilter (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if Analog Noise Filter is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 ANFOFF LL_FMPI2C_IsEnabledAnalogFilter

LL_FMPI2C_EnableDMAReq_TX

Function name

__STATIC_INLINE void LL_FMPI2C_EnableDMAReq_TX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable DMA transmission requests.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TXDMAEN LL_FMPI2C_EnableDMAReq_TX

LL_FMPI2C_DisableDMAReq_TX

Function name

__STATIC_INLINE void LL_FMPI2C_DisableDMAReq_TX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable DMA transmission requests.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TXDMAEN LL_FMPI2C_DisableDMAReq_TX

LL_FMPI2C_IsEnabledDMAReq_TX

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledDMAReq_TX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if DMA transmission requests are enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 TXDMAEN LL_FMPI2C_IsEnabledDMAReq_TX

LL_FMPI2C_EnableDMAReq_RX

Function name

__STATIC_INLINE void LL_FMPI2C_EnableDMAReq_RX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable DMA reception requests.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RXDMAEN LL_FMPI2C_EnableDMAReq_RX

LL_FMPI2C_DisableDMAReq_RX

Function name

__STATIC_INLINE void LL_FMPI2C_DisableDMAReq_RX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable DMA reception requests.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RXDMAEN LL_FMPI2C_DisableDMAReq_RX

LL_FMPI2C_IsEnabledDMAReq_RX

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledDMAReq_RX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if DMA reception requests are enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 RXDMAEN LL_FMPI2C_IsEnabledDMAReq_RX

LL_FMPI2C_DMA_GetRegAddr

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_DMA_GetRegAddr (FMPI2C_TypeDef * FMPI2Cx, uint32_t Direction)

Function description

Get the data register address used for DMA transfer.

Parameters

- **FMPI2Cx:** FMPI2C Instance
- **Direction:** This parameter can be one of the following values:
  - LL_FMPI2C_DMA_REG_DATA_TRANSMIT
  - LL_FMPI2C_DMA_REG_DATA_RECEIVE

Return values

- **Address:** of data register

Reference Manual to LL API cross reference:

- TXDR TXDATA LL_FMPI2C_DMA_GetRegAddr
- RXDR RXDATA LL_FMPI2C_DMA_GetRegAddr

LL_FMPI2C_EnableClockStretching

Function name

__STATIC_INLINE void LL_FMPI2C_EnableClockStretching (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable Clock stretching.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- This bit can only be programmed when the FMPI2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR1 NOSTRETCH LL_FMPI2C_EnableClockStretching

LL_FMPI2C_DisableClockStretching

Function name

__STATIC_INLINE void LL_FMPI2C_DisableClockStretching (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable Clock stretching.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- This bit can only be programmed when the FMPI2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR1 NOSTRETCH LL_FMPI2C_DisableClockStretching

LL_FMPI2C_IsEnabledClockStretching

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledClockStretching (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if Clock stretching is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 NOSTRETCH LL_FMPI2C_IsEnabledClockStretching

LL_FMPI2C_EnableSlaveByteControl

Function name

__STATIC_INLINE void LL_FMPI2C_EnableSlaveByteControl (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable hardware byte control in slave mode.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 SBC LL_FMPI2C_EnableSlaveByteControl

LL_FMPI2C_DisableSlaveByteControl

Function name

__STATIC_INLINE void LL_FMPI2C_DisableSlaveByteControl (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable hardware byte control in slave mode.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 SBC LL_FMPI2C_DisableSlaveByteControl

LL_FMPI2C_IsEnabledSlaveByteControl

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledSlaveByteControl (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if hardware byte control in slave mode is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 SBC LL_FMPI2C_IsEnabledSlaveByteControl

LL_FMPI2C_EnableGeneralCall

Function name

__STATIC_INLINE void LL_FMPI2C_EnableGeneralCall (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable General Call.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- When enabled the Address 0x00 is ACKed.

Reference Manual to LL API cross reference:

- CR1 GCEN LL_FMPI2C_EnableGeneralCall

LL_FMPI2C_DisableGeneralCall

Function name

__STATIC_INLINE void LL_FMPI2C_DisableGeneralCall (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable General Call.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- When disabled the Address 0x00 is NACKed.

Reference Manual to LL API cross reference:

- CR1 GCEN LL_FMPI2C_DisableGeneralCall

LL_FMPI2C_IsEnabledGeneralCall

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledGeneralCall (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if General Call is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 GCEN LL_FMPI2C_IsEnabledGeneralCall

LL_FMPI2C_SetMasterAddressingMode

Function name

__STATIC_INLINE void LL_FMPI2C_SetMasterAddressingMode (FMPI2C_TypeDef * FMPI2Cx, uint32_t AddressingMode)

Function description

Configure the Master to operate in 7-bit or 10-bit addressing mode.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **AddressingMode:** This parameter can be one of the following values:
  - LL_FMPI2C_ADDRESSING_MODE_7BIT
  - LL_FMPI2C_ADDRESSING_MODE_10BIT

Return values

- **None:**

Notes

- Changing this bit is not allowed, when the START bit is set.

Reference Manual to LL API cross reference:

- CR2 ADD10 LL_FMPI2C_SetMasterAddressingMode

LL_FMPI2C_GetMasterAddressingMode

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetMasterAddressingMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the Master addressing mode.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_FMPI2C_ADDRESSING_MODE_7BIT
  - LL_FMPI2C_ADDRESSING_MODE_10BIT

Reference Manual to LL API cross reference:

- CR2 ADD10 LL_FMPI2C_GetMasterAddressingMode

LL_FMPI2C_SetOwnAddress1

Function name

__STATIC_INLINE void LL_FMPI2C_SetOwnAddress1 (FMPI2C_TypeDef * FMPI2Cx, uint32_t OwnAddress1, uint32_t OwnAddrSize)

Function description

Set the Own Address1.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **OwnAddress1:** This parameter must be a value between Min_Data=0 and Max_Data=0x3FF.
- **OwnAddrSize:** This parameter can be one of the following values:
  - LL_FMPI2C_OWNADDRESS1_7BIT
  - LL_FMPI2C_OWNADDRESS1_10BIT

Return values

- **None:**

Reference Manual to LL API cross reference:

- OAR1 OA1 LL_FMPI2C_SetOwnAddress1
- OAR1 OA1MODE LL_FMPI2C_SetOwnAddress1

LL_FMPI2C_EnableOwnAddress1

Function name

__STATIC_INLINE void LL_FMPI2C_EnableOwnAddress1 (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable acknowledge on Own Address1 match address.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- OAR1 OA1EN LL_FMPI2C_EnableOwnAddress1

LL_FMPI2C_DisableOwnAddress1

Function name

__STATIC_INLINE void LL_FMPI2C_DisableOwnAddress1 (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable acknowledge on Own Address1 match address.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- OAR1 OA1EN LL_FMPI2C_DisableOwnAddress1

LL_FMPI2C_IsEnabledOwnAddress1

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledOwnAddress1 (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if Own Address1 acknowledge is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- OAR1 OA1EN LL_FMPI2C_IsEnabledOwnAddress1

LL_FMPI2C_SetOwnAddress2

Function name

__STATIC_INLINE void LL_FMPI2C_SetOwnAddress2 (FMPI2C_TypeDef * FMPI2Cx, uint32_t OwnAddress2, uint32_t OwnAddrMask)

Function description

Set the 7bits Own Address2.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **OwnAddress2:** Value between Min_Data=0 and Max_Data=0x7F.
- **OwnAddrMask:** This parameter can be one of the following values:
  - LL_FMPI2C_OWNADDRESS2_NOMASK
  - LL_FMPI2C_OWNADDRESS2_MASK01
  - LL_FMPI2C_OWNADDRESS2_MASK02
  - LL_FMPI2C_OWNADDRESS2_MASK03
  - LL_FMPI2C_OWNADDRESS2_MASK04
  - LL_FMPI2C_OWNADDRESS2_MASK05
  - LL_FMPI2C_OWNADDRESS2_MASK06
  - LL_FMPI2C_OWNADDRESS2_MASK07

Return values

- **None:**

Notes

- This action has no effect if own address2 is enabled.

Reference Manual to LL API cross reference:

- OAR2 OA2 LL_FMPI2C_SetOwnAddress2
- OAR2 OA2MSK LL_FMPI2C_SetOwnAddress2

LL_FMPI2C_EnableOwnAddress2

Function name

__STATIC_INLINE void LL_FMPI2C_EnableOwnAddress2 (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable acknowledge on Own Address2 match address.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- OAR2 OA2EN LL_FMPI2C_EnableOwnAddress2

LL_FMPI2C_DisableOwnAddress2

Function name

__STATIC_INLINE void LL_FMPI2C_DisableOwnAddress2 (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable acknowledge on Own Address2 match address.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- OAR2 OA2EN LL_FMPI2C_DisableOwnAddress2

LL_FMPI2C_IsEnabledOwnAddress2

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledOwnAddress2 (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if Own Address1 acknowledge is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- OAR2 OA2EN LL_FMPI2C_IsEnabledOwnAddress2

LL_FMPI2C_SetTiming

Function name

__STATIC_INLINE void LL_FMPI2C_SetTiming (FMPI2C_TypeDef * FMPI2Cx, uint32_t Timing)

Function description

Configure the SDA setup, hold time and the SCL high, low period.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **Timing:** This parameter must be a value between Min_Data=0 and Max_Data=0xFFFFFFFF.

Return values

- **None:**

Notes

- This bit can only be programmed when the FMPI2C is disabled (PE = 0).
- This parameter is computed with the STM32CubeMX Tool.

Reference Manual to LL API cross reference:

- TIMINGR TIMINGR LL_FMPI2C_SetTiming

LL_FMPI2C_GetTimingPrescaler

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetTimingPrescaler (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the Timing Prescaler setting.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x0 and Max_Data=0xF

Reference Manual to LL API cross reference:

- TIMINGR PRESC LL_FMPI2C_GetTimingPrescaler

LL_FMPI2C_GetClockLowPeriod

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetClockLowPeriod (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the SCL low period setting.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- TIMINGR SCLL LL_FMPI2C_GetClockLowPeriod

LL_FMPI2C_GetClockHighPeriod

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetClockHighPeriod (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the SCL high period setting.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- TIMINGR SCLH LL_FMPI2C_GetClockHighPeriod

LL_FMPI2C_GetDataHoldTime

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetDataHoldTime (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the SDA hold time.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x0 and Max_Data=0xF

Reference Manual to LL API cross reference:

- TIMINGR SDADEL LL_FMPI2C_GetDataHoldTime

LL_FMPI2C_GetDataSetupTime

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetDataSetupTime (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the SDA setup time.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x0 and Max_Data=0xF

Reference Manual to LL API cross reference:

- TIMINGR SCLDEL LL_FMPI2C_GetDataSetupTime

LL_FMPI2C_SetMode

Function name

__STATIC_INLINE void LL_FMPI2C_SetMode (FMPI2C_TypeDef * FMPI2Cx, uint32_t PeripheralMode)

Function description

Configure peripheral mode.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **PeripheralMode:** This parameter can be one of the following values:
  - LL_FMPI2C_MODE_I2C
  - LL_FMPI2C_MODE_SMBUS_HOST
  - LL_FMPI2C_MODE_SMBUS_DEVICE
  - LL_FMPI2C_MODE_SMBUS_DEVICE_ARP

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 SMBHEN LL_FMPI2C_SetMode
- CR1 SMBDEN LL_FMPI2C_SetMode

LL_FMPI2C_GetMode

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get peripheral mode.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_FMPI2C_MODE_I2C
  - LL_FMPI2C_MODE_SMBUS_HOST
  - LL_FMPI2C_MODE_SMBUS_DEVICE
  - LL_FMPI2C_MODE_SMBUS_DEVICE_ARP

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 SMBHEN LL_FMPI2C_GetMode
- CR1 SMBDEN LL_FMPI2C_GetMode

LL_FMPI2C_EnableSMBusAlert

Function name

__STATIC_INLINE void LL_FMPI2C_EnableSMBusAlert (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable SMBus alert (Host or Device mode)

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- SMBus Device mode: SMBus Alert pin is drived low and Alert Response Address Header acknowledge is enabled. SMBus Host mode:SMBus Alert pin management is supported.

Reference Manual to LL API cross reference:

- CR1 ALERTEN LL_FMPI2C_EnableSMBusAlert

LL_FMPI2C_DisableSMBusAlert

Function name

__STATIC_INLINE void LL_FMPI2C_DisableSMBusAlert (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable SMBus alert (Host or Device mode)

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- SMBus Device mode: SMBus Alert pin is not drived (can be used as a standard GPIO) and Alert Response Address Header acknowledge is disabled. SMBus Host mode:SMBus Alert pin management is not supported.

Reference Manual to LL API cross reference:

- CR1 ALERTEN LL_FMPI2C_DisableSMBusAlert

LL_FMPI2C_IsEnabledSMBusAlert

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledSMBusAlert (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if SMBus alert (Host or Device mode) is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 ALERTEN LL_FMPI2C_IsEnabledSMBusAlert

LL_FMPI2C_EnableSMBusPEC

Function name

__STATIC_INLINE void LL_FMPI2C_EnableSMBusPEC (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable SMBus Packet Error Calculation (PEC).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 PECEN LL_FMPI2C_EnableSMBusPEC

LL_FMPI2C_DisableSMBusPEC

Function name

__STATIC_INLINE void LL_FMPI2C_DisableSMBusPEC (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable SMBus Packet Error Calculation (PEC).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 PECEN LL_FMPI2C_DisableSMBusPEC

LL_FMPI2C_IsEnabledSMBusPEC

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledSMBusPEC (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if SMBus Packet Error Calculation (PEC) is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 PECEN LL_FMPI2C_IsEnabledSMBusPEC

LL_FMPI2C_ConfigSMBusTimeout

Function name

__STATIC_INLINE void LL_FMPI2C_ConfigSMBusTimeout (FMPI2C_TypeDef * FMPI2Cx, uint32_t TimeoutA, uint32_t TimeoutAMode, uint32_t TimeoutB)

Function description

Configure the SMBus Clock Timeout.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **TimeoutA:** This parameter must be a value between Min_Data=0 and Max_Data=0xFFF.
- **TimeoutAMode:** This parameter can be one of the following values:
  - LL_FMPI2C_SMBUS_TIMEOUTA_MODE_SCL_LOW
  - LL_FMPI2C_SMBUS_TIMEOUTA_MODE_SDA_SCL_HIGH
- **TimeoutB:**

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- This configuration can only be programmed when associated Timeout is disabled (TimeoutA and/ orTimeoutB).

Reference Manual to LL API cross reference:

- TIMEOUTR TIMEOUTA LL_FMPI2C_ConfigSMBusTimeout
- TIMEOUTR TIDLE LL_FMPI2C_ConfigSMBusTimeout
- TIMEOUTR TIMEOUTB LL_FMPI2C_ConfigSMBusTimeout

LL_FMPI2C_SetSMBusTimeoutA

Function name

__STATIC_INLINE void LL_FMPI2C_SetSMBusTimeoutA (FMPI2C_TypeDef * FMPI2Cx, uint32_t TimeoutA)

Function description

Configure the SMBus Clock TimeoutA (SCL low timeout or SCL and SDA high timeout depends on TimeoutA mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **TimeoutA:** This parameter must be a value between Min_Data=0 and Max_Data=0xFFF.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- These bits can only be programmed when TimeoutA is disabled.

Reference Manual to LL API cross reference:

- TIMEOUTR TIMEOUTA LL_FMPI2C_SetSMBusTimeoutA

LL_FMPI2C_GetSMBusTimeoutA

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetSMBusTimeoutA (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the SMBus Clock TimeoutA setting.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0 and Max_Data=0xFFF

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- TIMEOUTR TIMEOUTA LL_FMPI2C_GetSMBusTimeoutA

LL_FMPI2C_SetSMBusTimeoutAMode

Function name

__STATIC_INLINE void LL_FMPI2C_SetSMBusTimeoutAMode (FMPI2C_TypeDef * FMPI2Cx, uint32_t TimeoutAMode)

Function description

Set the SMBus Clock TimeoutA mode.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **TimeoutAMode:** This parameter can be one of the following values:
  - LL_FMPI2C_SMBUS_TIMEOUTA_MODE_SCL_LOW
  - LL_FMPI2C_SMBUS_TIMEOUTA_MODE_SDA_SCL_HIGH

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- This bit can only be programmed when TimeoutA is disabled.

Reference Manual to LL API cross reference:

- TIMEOUTR TIDLE LL_FMPI2C_SetSMBusTimeoutAMode

LL_FMPI2C_GetSMBusTimeoutAMode

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetSMBusTimeoutAMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the SMBus Clock TimeoutA mode.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_FMPI2C_SMBUS_TIMEOUTA_MODE_SCL_LOW
  - LL_FMPI2C_SMBUS_TIMEOUTA_MODE_SDA_SCL_HIGH

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- TIMEOUTR TIDLE LL_FMPI2C_GetSMBusTimeoutAMode

LL_FMPI2C_SetSMBusTimeoutB

Function name

__STATIC_INLINE void LL_FMPI2C_SetSMBusTimeoutB (FMPI2C_TypeDef * FMPI2Cx, uint32_t TimeoutB)

Function description

Configure the SMBus Extended Cumulative Clock TimeoutB (Master or Slave mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **TimeoutB:** This parameter must be a value between Min_Data=0 and Max_Data=0xFFF.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- These bits can only be programmed when TimeoutB is disabled.

Reference Manual to LL API cross reference:

- TIMEOUTR TIMEOUTB LL_FMPI2C_SetSMBusTimeoutB

LL_FMPI2C_GetSMBusTimeoutB

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetSMBusTimeoutB (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the SMBus Extented Cumulative Clock TimeoutB setting.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0 and Max_Data=0xFFF

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- TIMEOUTR TIMEOUTB LL_FMPI2C_GetSMBusTimeoutB

LL_FMPI2C_EnableSMBusTimeout

Function name

__STATIC_INLINE void LL_FMPI2C_EnableSMBusTimeout (FMPI2C_TypeDef * FMPI2Cx, uint32_t ClockTimeout)

Function description

Enable the SMBus Clock Timeout.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **ClockTimeout:** This parameter can be one of the following values:
  - LL_FMPI2C_SMBUS_TIMEOUTA
  - LL_FMPI2C_SMBUS_TIMEOUTB
  - LL_FMPI2C_SMBUS_ALL_TIMEOUT

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- TIMEOUTR TIMOUTEN LL_FMPI2C_EnableSMBusTimeout
- TIMEOUTR TEXTEN LL_FMPI2C_EnableSMBusTimeout

LL_FMPI2C_DisableSMBusTimeout

Function name

__STATIC_INLINE void LL_FMPI2C_DisableSMBusTimeout (FMPI2C_TypeDef * FMPI2Cx, uint32_t ClockTimeout)

Function description

Disable the SMBus Clock Timeout.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **ClockTimeout:** This parameter can be one of the following values:
  - LL_FMPI2C_SMBUS_TIMEOUTA
  - LL_FMPI2C_SMBUS_TIMEOUTB
  - LL_FMPI2C_SMBUS_ALL_TIMEOUT

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- TIMEOUTR TIMOUTEN LL_FMPI2C_DisableSMBusTimeout
- TIMEOUTR TEXTEN LL_FMPI2C_DisableSMBusTimeout

LL_FMPI2C_IsEnabledSMBusTimeout

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledSMBusTimeout (FMPI2C_TypeDef * FMPI2Cx, uint32_t ClockTimeout)

Function description

Check if the SMBus Clock Timeout is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **ClockTimeout:** This parameter can be one of the following values:
  - LL_FMPI2C_SMBUS_TIMEOUTA
  - LL_FMPI2C_SMBUS_TIMEOUTB
  - LL_FMPI2C_SMBUS_ALL_TIMEOUT

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- TIMEOUTR TIMOUTEN LL_FMPI2C_IsEnabledSMBusTimeout
- TIMEOUTR TEXTEN LL_FMPI2C_IsEnabledSMBusTimeout

LL_FMPI2C_EnableIT_TX

Function name

__STATIC_INLINE void LL_FMPI2C_EnableIT_TX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable TXIS interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TXIE LL_FMPI2C_EnableIT_TX

LL_FMPI2C_DisableIT_TX

Function name

__STATIC_INLINE void LL_FMPI2C_DisableIT_TX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable TXIS interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TXIE LL_FMPI2C_DisableIT_TX

LL_FMPI2C_IsEnabledIT_TX

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledIT_TX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if the TXIS Interrupt is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 TXIE LL_FMPI2C_IsEnabledIT_TX

LL_FMPI2C_EnableIT_RX

Function name

__STATIC_INLINE void LL_FMPI2C_EnableIT_RX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable RXNE interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RXIE LL_FMPI2C_EnableIT_RX

LL_FMPI2C_DisableIT_RX

Function name

__STATIC_INLINE void LL_FMPI2C_DisableIT_RX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable RXNE interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RXIE LL_FMPI2C_DisableIT_RX

LL_FMPI2C_IsEnabledIT_RX

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledIT_RX (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if the RXNE Interrupt is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 RXIE LL_FMPI2C_IsEnabledIT_RX

LL_FMPI2C_EnableIT_ADDR

Function name

__STATIC_INLINE void LL_FMPI2C_EnableIT_ADDR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable Address match interrupt (slave mode only).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 ADDRIE LL_FMPI2C_EnableIT_ADDR

LL_FMPI2C_DisableIT_ADDR

Function name

__STATIC_INLINE void LL_FMPI2C_DisableIT_ADDR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable Address match interrupt (slave mode only).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 ADDRIE LL_FMPI2C_DisableIT_ADDR

LL_FMPI2C_IsEnabledIT_ADDR

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledIT_ADDR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if Address match interrupt is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 ADDRIE LL_FMPI2C_IsEnabledIT_ADDR

LL_FMPI2C_EnableIT_NACK

Function name

__STATIC_INLINE void LL_FMPI2C_EnableIT_NACK (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable Not acknowledge received interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 NACKIE LL_FMPI2C_EnableIT_NACK

LL_FMPI2C_DisableIT_NACK

Function name

__STATIC_INLINE void LL_FMPI2C_DisableIT_NACK (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable Not acknowledge received interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 NACKIE LL_FMPI2C_DisableIT_NACK

LL_FMPI2C_IsEnabledIT_NACK

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledIT_NACK (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if Not acknowledge received interrupt is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 NACKIE LL_FMPI2C_IsEnabledIT_NACK

LL_FMPI2C_EnableIT_STOP

Function name

__STATIC_INLINE void LL_FMPI2C_EnableIT_STOP (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable STOP detection interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 STOPIE LL_FMPI2C_EnableIT_STOP

LL_FMPI2C_DisableIT_STOP

Function name

__STATIC_INLINE void LL_FMPI2C_DisableIT_STOP (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable STOP detection interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 STOPIE LL_FMPI2C_DisableIT_STOP

LL_FMPI2C_IsEnabledIT_STOP

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledIT_STOP (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if STOP detection interrupt is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 STOPIE LL_FMPI2C_IsEnabledIT_STOP

LL_FMPI2C_EnableIT_TC

Function name

__STATIC_INLINE void LL_FMPI2C_EnableIT_TC (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable Transfer Complete interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Any of these events will generate interrupt : Transfer Complete (TC) Transfer Complete Reload (TCR)

Reference Manual to LL API cross reference:

- CR1 TCIE LL_FMPI2C_EnableIT_TC

LL_FMPI2C_DisableIT_TC

Function name

__STATIC_INLINE void LL_FMPI2C_DisableIT_TC (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable Transfer Complete interrupt.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Any of these events will generate interrupt : Transfer Complete (TC) Transfer Complete Reload (TCR)

Reference Manual to LL API cross reference:

- CR1 TCIE LL_FMPI2C_DisableIT_TC

LL_FMPI2C_IsEnabledIT_TC

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledIT_TC (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if Transfer Complete interrupt is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 TCIE LL_FMPI2C_IsEnabledIT_TC

LL_FMPI2C_EnableIT_ERR

Function name

__STATIC_INLINE void LL_FMPI2C_EnableIT_ERR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable Error interrupts.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- Any of these errors will generate interrupt : Arbitration Loss (ARLO) Bus Error detection (BERR) Overrun/ Underrun (OVR) SMBus Timeout detection (TIMEOUT) SMBus PEC error detection (PECERR) SMBus Alert pin event detection (ALERT)

Reference Manual to LL API cross reference:

- CR1 ERRIE LL_FMPI2C_EnableIT_ERR

LL_FMPI2C_DisableIT_ERR

Function name

__STATIC_INLINE void LL_FMPI2C_DisableIT_ERR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable Error interrupts.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- Any of these errors will generate interrupt : Arbitration Loss (ARLO) Bus Error detection (BERR) Overrun/ Underrun (OVR) SMBus Timeout detection (TIMEOUT) SMBus PEC error detection (PECERR) SMBus Alert pin event detection (ALERT)

Reference Manual to LL API cross reference:

- CR1 ERRIE LL_FMPI2C_DisableIT_ERR

LL_FMPI2C_IsEnabledIT_ERR

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledIT_ERR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if Error interrupts are enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 ERRIE LL_FMPI2C_IsEnabledIT_ERR

LL_FMPI2C_IsActiveFlag_TXE

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_TXE (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Transmit data register empty flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: When next data is written in Transmit data register. SET: When Transmit data register is empty.

Reference Manual to LL API cross reference:

- ISR TXE LL_FMPI2C_IsActiveFlag_TXE

LL_FMPI2C_IsActiveFlag_TXIS

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_TXIS (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Transmit interrupt flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: When next data is written in Transmit data register. SET: When Transmit data register is empty.

Reference Manual to LL API cross reference:

- ISR TXIS LL_FMPI2C_IsActiveFlag_TXIS

LL_FMPI2C_IsActiveFlag_RXNE

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_RXNE (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Receive data register not empty flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: When Receive data register is read. SET: When the received data is copied in Receive data register.

Reference Manual to LL API cross reference:

- ISR RXNE LL_FMPI2C_IsActiveFlag_RXNE

LL_FMPI2C_IsActiveFlag_ADDR

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_ADDR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Address matched flag (slave mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When the received slave address matched with one of the enabled slave address.

Reference Manual to LL API cross reference:

- ISR ADDR LL_FMPI2C_IsActiveFlag_ADDR

LL_FMPI2C_IsActiveFlag_NACK

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_NACK (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Not Acknowledge received flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When a NACK is received after a byte transmission.

Reference Manual to LL API cross reference:

- ISR NACKF LL_FMPI2C_IsActiveFlag_NACK

LL_FMPI2C_IsActiveFlag_STOP

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_STOP (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Stop detection flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When a Stop condition is detected.

Reference Manual to LL API cross reference:

- ISR STOPF LL_FMPI2C_IsActiveFlag_STOP

LL_FMPI2C_IsActiveFlag_TC

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_TC (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Transfer complete flag (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When RELOAD=0, AUTOEND=0 and NBYTES date have been transferred.

Reference Manual to LL API cross reference:

- ISR TC LL_FMPI2C_IsActiveFlag_TC

LL_FMPI2C_IsActiveFlag_TCR

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_TCR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Transfer complete flag (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When RELOAD=1 and NBYTES date have been transferred.

Reference Manual to LL API cross reference:

- ISR TCR LL_FMPI2C_IsActiveFlag_TCR

LL_FMPI2C_IsActiveFlag_BERR

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_BERR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Bus error flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When a misplaced Start or Stop condition is detected.

Reference Manual to LL API cross reference:

- ISR BERR LL_FMPI2C_IsActiveFlag_BERR

LL_FMPI2C_IsActiveFlag_ARLO

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_ARLO (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Arbitration lost flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When arbitration lost.

Reference Manual to LL API cross reference:

- ISR ARLO LL_FMPI2C_IsActiveFlag_ARLO

LL_FMPI2C_IsActiveFlag_OVR

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_OVR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Overrun/Underrun flag (slave mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When an overrun/underrun error occurs (Clock Stretching Disabled).

Reference Manual to LL API cross reference:

- ISR OVR LL_FMPI2C_IsActiveFlag_OVR

LL_FMPI2C_IsActiveSMBusFlag_PECERR

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveSMBusFlag_PECERR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of SMBus PEC error flag in reception.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- RESET: Clear default value. SET: When the received PEC does not match with the PEC register content.

Reference Manual to LL API cross reference:

- ISR PECERR LL_FMPI2C_IsActiveSMBusFlag_PECERR

LL_FMPI2C_IsActiveSMBusFlag_TIMEOUT

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveSMBusFlag_TIMEOUT (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of SMBus Timeout detection flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- RESET: Clear default value. SET: When a timeout or extended clock timeout occurs.

Reference Manual to LL API cross reference:

- ISR TIMEOUT LL_FMPI2C_IsActiveSMBusFlag_TIMEOUT

LL_FMPI2C_IsActiveSMBusFlag_ALERT

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveSMBusFlag_ALERT (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of SMBus alert flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- RESET: Clear default value. SET: When SMBus host configuration, SMBus alert enabled and a falling edge event occurs on SMBA pin.

Reference Manual to LL API cross reference:

- ISR ALERT LL_FMPI2C_IsActiveSMBusFlag_ALERT

LL_FMPI2C_IsActiveFlag_BUSY

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsActiveFlag_BUSY (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the status of Bus Busy flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When a Start condition is detected.

Reference Manual to LL API cross reference:

- ISR BUSY LL_FMPI2C_IsActiveFlag_BUSY

LL_FMPI2C_ClearFlag_ADDR

Function name

__STATIC_INLINE void LL_FMPI2C_ClearFlag_ADDR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear Address Matched flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR ADDRCF LL_FMPI2C_ClearFlag_ADDR

LL_FMPI2C_ClearFlag_NACK

Function name

__STATIC_INLINE void LL_FMPI2C_ClearFlag_NACK (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear Not Acknowledge flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR NACKCF LL_FMPI2C_ClearFlag_NACK

LL_FMPI2C_ClearFlag_STOP

Function name

__STATIC_INLINE void LL_FMPI2C_ClearFlag_STOP (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear Stop detection flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR STOPCF LL_FMPI2C_ClearFlag_STOP

LL_FMPI2C_ClearFlag_TXE

Function name

__STATIC_INLINE void LL_FMPI2C_ClearFlag_TXE (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear Transmit data register empty flag (TXE).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- This bit can be clear by software in order to flush the transmit data register (TXDR).

Reference Manual to LL API cross reference:

- ISR TXE LL_FMPI2C_ClearFlag_TXE

LL_FMPI2C_ClearFlag_BERR

Function name

__STATIC_INLINE void LL_FMPI2C_ClearFlag_BERR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear Bus error flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR BERRCF LL_FMPI2C_ClearFlag_BERR

LL_FMPI2C_ClearFlag_ARLO

Function name

__STATIC_INLINE void LL_FMPI2C_ClearFlag_ARLO (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear Arbitration lost flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR ARLOCF LL_FMPI2C_ClearFlag_ARLO

LL_FMPI2C_ClearFlag_OVR

Function name

__STATIC_INLINE void LL_FMPI2C_ClearFlag_OVR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear Overrun/Underrun flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR OVRCF LL_FMPI2C_ClearFlag_OVR

LL_FMPI2C_ClearSMBusFlag_PECERR

Function name

__STATIC_INLINE void LL_FMPI2C_ClearSMBusFlag_PECERR (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear SMBus PEC error flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- ICR PECCF LL_FMPI2C_ClearSMBusFlag_PECERR

LL_FMPI2C_ClearSMBusFlag_TIMEOUT

Function name

__STATIC_INLINE void LL_FMPI2C_ClearSMBusFlag_TIMEOUT (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear SMBus Timeout detection flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- ICR TIMOUTCF LL_FMPI2C_ClearSMBusFlag_TIMEOUT

LL_FMPI2C_ClearSMBusFlag_ALERT

Function name

__STATIC_INLINE void LL_FMPI2C_ClearSMBusFlag_ALERT (FMPI2C_TypeDef * FMPI2Cx)

Function description

Clear SMBus Alert flag.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- ICR ALERTCF LL_FMPI2C_ClearSMBusFlag_ALERT

LL_FMPI2C_EnableAutoEndMode

Function name

__STATIC_INLINE void LL_FMPI2C_EnableAutoEndMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable automatic STOP condition generation (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Automatic end mode : a STOP condition is automatically sent when NBYTES data are transferred. This bit has no effect in slave mode or when RELOAD bit is set.

Reference Manual to LL API cross reference:

- CR2 AUTOEND LL_FMPI2C_EnableAutoEndMode

LL_FMPI2C_DisableAutoEndMode

Function name

__STATIC_INLINE void LL_FMPI2C_DisableAutoEndMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable automatic STOP condition generation (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Software end mode : TC flag is set when NBYTES data are transferre, stretching SCL low.

Reference Manual to LL API cross reference:

- CR2 AUTOEND LL_FMPI2C_DisableAutoEndMode

LL_FMPI2C_IsEnabledAutoEndMode

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledAutoEndMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if automatic STOP condition is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 AUTOEND LL_FMPI2C_IsEnabledAutoEndMode

LL_FMPI2C_EnableReloadMode

Function name

__STATIC_INLINE void LL_FMPI2C_EnableReloadMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable reload mode (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- The transfer is not completed after the NBYTES data transfer, NBYTES will be reloaded when TCR flag is set.

Reference Manual to LL API cross reference:

- CR2 RELOAD LL_FMPI2C_EnableReloadMode

LL_FMPI2C_DisableReloadMode

Function name

__STATIC_INLINE void LL_FMPI2C_DisableReloadMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable reload mode (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- The transfer is completed after the NBYTES data transfer(STOP or RESTART will follow).

Reference Manual to LL API cross reference:

- CR2 RELOAD LL_FMPI2C_DisableReloadMode

LL_FMPI2C_IsEnabledReloadMode

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledReloadMode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if reload mode is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 RELOAD LL_FMPI2C_IsEnabledReloadMode

LL_FMPI2C_SetTransferSize

Function name

__STATIC_INLINE void LL_FMPI2C_SetTransferSize (FMPI2C_TypeDef * FMPI2Cx, uint32_t TransferSize)

Function description

Configure the number of bytes for transfer.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **TransferSize:** This parameter must be a value between Min_Data=0x00 and Max_Data=0xFF.

Return values

- **None:**

Notes

- Changing these bits when START bit is set is not allowed.

Reference Manual to LL API cross reference:

- CR2 NBYTES LL_FMPI2C_SetTransferSize

LL_FMPI2C_GetTransferSize

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetTransferSize (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the number of bytes configured for transfer.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- CR2 NBYTES LL_FMPI2C_GetTransferSize

LL_FMPI2C_AcknowledgeNextData

Function name

__STATIC_INLINE void LL_FMPI2C_AcknowledgeNextData (FMPI2C_TypeDef * FMPI2Cx, uint32_t TypeAcknowledge)

Function description

Prepare the generation of a ACKnowledge or Non ACKnowledge condition after the address receive match code or next received byte.

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **TypeAcknowledge:** This parameter can be one of the following values:
  - LL_FMPI2C_ACK
  - LL_FMPI2C_NACK

Return values

- **None:**

Notes

- Usage in Slave mode only.

Reference Manual to LL API cross reference:

- CR2 NACK LL_FMPI2C_AcknowledgeNextData

LL_FMPI2C_GenerateStartCondition

Function name

__STATIC_INLINE void LL_FMPI2C_GenerateStartCondition (FMPI2C_TypeDef * FMPI2Cx)

Function description

Generate a START or RESTART condition.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- The START bit can be set even if bus is BUSY or FMPI2C is in slave mode. This action has no effect when RELOAD is set.

Reference Manual to LL API cross reference:

- CR2 START LL_FMPI2C_GenerateStartCondition

LL_FMPI2C_GenerateStopCondition

Function name

__STATIC_INLINE void LL_FMPI2C_GenerateStopCondition (FMPI2C_TypeDef * FMPI2Cx)

Function description

Generate a STOP condition after the current byte transfer (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 STOP LL_FMPI2C_GenerateStopCondition

LL_FMPI2C_EnableAuto10BitRead

Function name

__STATIC_INLINE void LL_FMPI2C_EnableAuto10BitRead (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable automatic RESTART Read request condition for 10bit address header (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- The master sends the complete 10bit slave address read sequence : Start + 2 bytes 10bit address in Write direction + Restart + first 7 bits of 10bit address in Read direction.

Reference Manual to LL API cross reference:

- CR2 HEAD10R LL_FMPI2C_EnableAuto10BitRead

LL_FMPI2C_DisableAuto10BitRead

Function name

__STATIC_INLINE void LL_FMPI2C_DisableAuto10BitRead (FMPI2C_TypeDef * FMPI2Cx)

Function description

Disable automatic RESTART Read request condition for 10bit address header (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- The master only sends the first 7 bits of 10bit address in Read direction.

Reference Manual to LL API cross reference:

- CR2 HEAD10R LL_FMPI2C_DisableAuto10BitRead

LL_FMPI2C_IsEnabledAuto10BitRead

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledAuto10BitRead (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if automatic RESTART Read request condition for 10bit address header is enabled or disabled.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 HEAD10R LL_FMPI2C_IsEnabledAuto10BitRead

LL_FMPI2C_SetTransferRequest

Function name

__STATIC_INLINE void LL_FMPI2C_SetTransferRequest (FMPI2C_TypeDef * FMPI2Cx, uint32_t TransferRequest)

Function description

Configure the transfer direction (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **TransferRequest:** This parameter can be one of the following values:
  - LL_FMPI2C_REQUEST_WRITE
  - LL_FMPI2C_REQUEST_READ

Return values

- **None:**

Notes

- Changing these bits when START bit is set is not allowed.

Reference Manual to LL API cross reference:

- CR2 RD_WRN LL_FMPI2C_SetTransferRequest

LL_FMPI2C_GetTransferRequest

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetTransferRequest (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the transfer direction requested (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_FMPI2C_REQUEST_WRITE
  - LL_FMPI2C_REQUEST_READ

Reference Manual to LL API cross reference:

- CR2 RD_WRN LL_FMPI2C_GetTransferRequest

LL_FMPI2C_SetSlaveAddr

Function name

__STATIC_INLINE void LL_FMPI2C_SetSlaveAddr (FMPI2C_TypeDef * FMPI2Cx, uint32_t SlaveAddr)

Function description

Configure the slave address for transfer (master mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **SlaveAddr:** This parameter must be a value between Min_Data=0x00 and Max_Data=0x3F.

Return values

- **None:**

Notes

- Changing these bits when START bit is set is not allowed.

Reference Manual to LL API cross reference:

- CR2 SADD LL_FMPI2C_SetSlaveAddr

LL_FMPI2C_GetSlaveAddr

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetSlaveAddr (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the slave address programmed for transfer.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x0 and Max_Data=0x3F

Reference Manual to LL API cross reference:

- CR2 SADD LL_FMPI2C_GetSlaveAddr

LL_FMPI2C_HandleTransfer

Function name

__STATIC_INLINE void LL_FMPI2C_HandleTransfer (FMPI2C_TypeDef * FMPI2Cx, uint32_t SlaveAddr, uint32_t SlaveAddrSize, uint32_t TransferSize, uint32_t EndMode, uint32_t Request)

Function description

Handles FMPI2Cx communication when starting transfer or during transfer (TC or TCR flag are set).

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **SlaveAddr:** Specifies the slave address to be programmed.
- **SlaveAddrSize:** This parameter can be one of the following values:
  - LL_FMPI2C_ADDRSLAVE_7BIT
  - LL_FMPI2C_ADDRSLAVE_10BIT
- **TransferSize:** Specifies the number of bytes to be programmed. This parameter must be a value between Min_Data=0 and Max_Data=255.
- **EndMode:** This parameter can be one of the following values:
  - LL_FMPI2C_MODE_RELOAD
  - LL_FMPI2C_MODE_AUTOEND
  - LL_FMPI2C_MODE_SOFTEND
  - LL_FMPI2C_MODE_SMBUS_RELOAD
  - LL_FMPI2C_MODE_SMBUS_AUTOEND_NO_PEC
  - LL_FMPI2C_MODE_SMBUS_SOFTEND_NO_PEC
  - LL_FMPI2C_MODE_SMBUS_AUTOEND_WITH_PEC
  - LL_FMPI2C_MODE_SMBUS_SOFTEND_WITH_PEC
- **Request:** This parameter can be one of the following values:
  - LL_FMPI2C_GENERATE_NOSTARTSTOP
  - LL_FMPI2C_GENERATE_STOP
  - LL_FMPI2C_GENERATE_START_READ
  - LL_FMPI2C_GENERATE_START_WRITE
  - LL_FMPI2C_GENERATE_RESTART_7BIT_READ
  - LL_FMPI2C_GENERATE_RESTART_7BIT_WRITE
  - LL_FMPI2C_GENERATE_RESTART_10BIT_READ
  - LL_FMPI2C_GENERATE_RESTART_10BIT_WRITE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 SADD LL_FMPI2C_HandleTransfer
- CR2 ADD10 LL_FMPI2C_HandleTransfer
- CR2 RD_WRN LL_FMPI2C_HandleTransfer
- CR2 START LL_FMPI2C_HandleTransfer
- CR2 STOP LL_FMPI2C_HandleTransfer
- CR2 RELOAD LL_FMPI2C_HandleTransfer
- CR2 NBYTES LL_FMPI2C_HandleTransfer
- CR2 AUTOEND LL_FMPI2C_HandleTransfer
- CR2 HEAD10R LL_FMPI2C_HandleTransfer

LL_FMPI2C_GetTransferDirection

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetTransferDirection (FMPI2C_TypeDef * FMPI2Cx)

Function description

Indicate the value of transfer direction (slave mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_FMPI2C_DIRECTION_WRITE
  - LL_FMPI2C_DIRECTION_READ

Notes

- RESET: Write transfer, Slave enters in receiver mode. SET: Read transfer, Slave enters in transmitter mode.

Reference Manual to LL API cross reference:

- ISR DIR LL_FMPI2C_GetTransferDirection

LL_FMPI2C_GetAddressMatchCode

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetAddressMatchCode (FMPI2C_TypeDef * FMPI2Cx)

Function description

Return the slave matched address.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x3F

Reference Manual to LL API cross reference:

- ISR ADDCODE LL_FMPI2C_GetAddressMatchCode

LL_FMPI2C_EnableSMBusPECCompare

Function name

__STATIC_INLINE void LL_FMPI2C_EnableSMBusPECCompare (FMPI2C_TypeDef * FMPI2Cx)

Function description

Enable internal comparison of the SMBus Packet Error byte (transmission or reception mode).

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **None:**

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.
- This feature is cleared by hardware when the PEC byte is transferred, or when a STOP condition or an Address Matched is received. This bit has no effect when RELOAD bit is set. This bit has no effect in device mode when SBC bit is not set.

Reference Manual to LL API cross reference:

- CR2 PECBYTE LL_FMPI2C_EnableSMBusPECCompare

LL_FMPI2C_IsEnabledSMBusPECCompare

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_IsEnabledSMBusPECCompare (FMPI2C_TypeDef * FMPI2Cx)

Function description

Check if the SMBus Packet Error byte internal comparison is requested or not.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- CR2 PECBYTE LL_FMPI2C_IsEnabledSMBusPECCompare

LL_FMPI2C_GetSMBusPEC

Function name

__STATIC_INLINE uint32_t LL_FMPI2C_GetSMBusPEC (FMPI2C_TypeDef * FMPI2Cx)

Function description

Get the SMBus Packet Error byte calculated.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Notes

- Macro IS_FMPSMBUS_ALL_INSTANCE(FMPI2Cx) can be used to check whether or not SMBus feature is supported by the FMPI2Cx Instance.

Reference Manual to LL API cross reference:

- PECR PEC LL_FMPI2C_GetSMBusPEC

LL_FMPI2C_ReceiveData8

Function name

__STATIC_INLINE uint8_t LL_FMPI2C_ReceiveData8 (FMPI2C_TypeDef * FMPI2Cx)

Function description

Read Receive Data register.

Parameters

- **FMPI2Cx:** FMPI2C Instance.

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- RXDR RXDATA LL_FMPI2C_ReceiveData8

LL_FMPI2C_TransmitData8

Function name

__STATIC_INLINE void LL_FMPI2C_TransmitData8 (FMPI2C_TypeDef * FMPI2Cx, uint8_t Data)

Function description

Write in Transmit Data Register .

Parameters

- **FMPI2Cx:** FMPI2C Instance.
- **Data:** Value between Min_Data=0x00 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- TXDR TXDATA LL_FMPI2C_TransmitData8

LL_FMPI2C_Init

Function name

ErrorStatus LL_FMPI2C_Init (FMPI2C_TypeDef * FMPI2Cx, LL_FMPI2C_InitTypeDef * FMPI2C_InitStruct)

Function description

LL_FMPI2C_DeInit

Function name

ErrorStatus LL_FMPI2C_DeInit (FMPI2C_TypeDef * FMPI2Cx)

Function description

LL_FMPI2C_StructInit

Function name

void LL_FMPI2C_StructInit (LL_FMPI2C_InitTypeDef * FMPI2C_InitStruct)

Function description

80.3 FMPI2C Firmware driver defines

The following section lists the various define and macros of the module.

80.3.1 FMPI2C

FMPI2C

Master Addressing Mode

LL_FMPI2C_ADDRESSING_MODE_7BIT

Master operates in 7-bit addressing mode.

LL_FMPI2C_ADDRESSING_MODE_10BIT

Master operates in 10-bit addressing mode.

Slave Address Length

LL_FMPI2C_ADDRSLAVE_7BIT

Slave Address in 7-bit.

LL_FMPI2C_ADDRSLAVE_10BIT

Slave Address in 10-bit.

Analog Filter Selection

LL_FMPI2C_ANALOGFILTER_ENABLE

Analog filter is enabled.

LL_FMPI2C_ANALOGFILTER_DISABLE

Analog filter is disabled.

Clear Flags Defines

LL_FMPI2C_ICR_ADDRCF

Address Matched flag

LL_FMPI2C_ICR_NACKCF

Not Acknowledge flag

LL_FMPI2C_ICR_STOPCF

Stop detection flag

LL_FMPI2C_ICR_BERRCF

Bus error flag

LL_FMPI2C_ICR_ARLOCF

Arbitration Lost flag

LL_FMPI2C_ICR_OVRCF

Overrun/Underrun flag

LL_FMPI2C_ICR_PECCF

PEC error flag

LL_FMPI2C_ICR_TIMOUTCF

Timeout detection flag

LL_FMPI2C_ICR_ALERTCF

Alert flag

Read Write Direction

LL_FMPI2C_DIRECTION_WRITE

Write transfer request by master, slave enters receiver mode.

LL_FMPI2C_DIRECTION_READ

Read transfer request by master, slave enters transmitter mode.

DMA Register Data

LL_FMPI2C_DMA_REG_DATA_TRANSMIT

Get address of data register used for transmission

LL_FMPI2C_DMA_REG_DATA_RECEIVE

Get address of data register used for reception

Start And Stop Generation

LL_FMPI2C_GENERATE_NOSTARTSTOP

Don't Generate Stop and Start condition.

LL_FMPI2C_GENERATE_STOP

Generate Stop condition (Size should be set to 0).

LL_FMPI2C_GENERATE_START_READ

Generate Start for read request.

LL_FMPI2C_GENERATE_START_WRITE

Generate Start for write request.

LL_FMPI2C_GENERATE_RESTART_7BIT_READ

Generate Restart for read request, slave 7Bit address.

LL_FMPI2C_GENERATE_RESTART_7BIT_WRITE

Generate Restart for write request, slave 7Bit address.

LL_FMPI2C_GENERATE_RESTART_10BIT_READ

Generate Restart for read request, slave 10Bit address.

LL_FMPI2C_GENERATE_RESTART_10BIT_WRITE

Generate Restart for write request, slave 10Bit address.

Get Flags Defines

LL_FMPI2C_ISR_TXE

Transmit data register empty

LL_FMPI2C_ISR_TXIS

Transmit interrupt status

LL_FMPI2C_ISR_RXNE

Receive data register not empty

LL_FMPI2C_ISR_ADDR

Address matched (slave mode)

LL_FMPI2C_ISR_NACKF

Not Acknowledge received flag

LL_FMPI2C_ISR_STOPF

Stop detection flag

LL_FMPI2C_ISR_TC

Transfer Complete (master mode)

LL_FMPI2C_ISR_TCR

Transfer Complete Reload

LL_FMPI2C_ISR_BERR

Bus error

LL_FMPI2C_ISR_ARLO

Arbitration lost

LL_FMPI2C_ISR_OVR

Overrun/Underrun (slave mode)

LL_FMPI2C_ISR_PECERR

PEC Error in reception (SMBus mode)

LL_FMPI2C_ISR_TIMEOUT

Timeout detection flag (SMBus mode)

LL_FMPI2C_ISR_ALERT

SMBus alert (SMBus mode)

LL_FMPI2C_ISR_BUSY

Bus busy

Acknowledge Generation

LL_FMPI2C_ACK

ACK is sent after current received byte.

LL_FMPI2C_NACK

NACK is sent after current received byte.

IT Defines

LL_FMPI2C_CR1_TXIE

TX Interrupt enable

LL_FMPI2C_CR1_RXIE

RX Interrupt enable

LL_FMPI2C_CR1_ADDRIE

Address match Interrupt enable (slave only)

LL_FMPI2C_CR1_NACKIE

Not acknowledge received Interrupt enable

LL_FMPI2C_CR1_STOPIE

STOP detection Interrupt enable

LL_FMPI2C_CR1_TCIE

Transfer Complete interrupt enable

LL_FMPI2C_CR1_ERRIE

Error interrupts enable

Transfer End Mode

LL_FMPI2C_MODE_RELOAD

Enable FMPI2C Reload mode.

LL_FMPI2C_MODE_AUTOEND

Enable FMPI2C Automatic end mode with no HW PEC comparison.

LL_FMPI2C_MODE_SOFTEND

Enable FMPI2C Software end mode with no HW PEC comparison.

LL_FMPI2C_MODE_SMBUS_RELOAD

Enable SMBUS Automatic end mode with HW PEC comparison.

LL_FMPI2C_MODE_SMBUS_AUTOEND_NO_PEC

Enable SMBUS Automatic end mode with HW PEC comparison.

LL_FMPI2C_MODE_SMBUS_SOFTEND_NO_PEC

Enable SMBUS Software end mode with HW PEC comparison.

LL_FMPI2C_MODE_SMBUS_AUTOEND_WITH_PEC

Enable SMBUS Automatic end mode with HW PEC comparison.

LL_FMPI2C_MODE_SMBUS_SOFTEND_WITH_PEC

Enable SMBUS Software end mode with HW PEC comparison.

Own Address 1 Length

LL_FMPI2C_OWNADDRESS1_7BIT

Own address 1 is a 7-bit address.

LL_FMPI2C_OWNADDRESS1_10BIT

Own address 1 is a 10-bit address.

Own Address 2 Masks

LL_FMPI2C_OWNADDRESS2_NOMASK

Own Address2 No mask.

LL_FMPI2C_OWNADDRESS2_MASK01

Only Address2 bits[7:2] are compared.

LL_FMPI2C_OWNADDRESS2_MASK02

Only Address2 bits[7:3] are compared.

LL_FMPI2C_OWNADDRESS2_MASK03

Only Address2 bits[7:4] are compared.

LL_FMPI2C_OWNADDRESS2_MASK04

Only Address2 bits[7:5] are compared.

LL_FMPI2C_OWNADDRESS2_MASK05

Only Address2 bits[7:6] are compared.

LL_FMPI2C_OWNADDRESS2_MASK06

Only Address2 bits[7] are compared.

LL_FMPI2C_OWNADDRESS2_MASK07

No comparison is done. All Address2 are acknowledged.

Peripheral Mode

LL_FMPI2C_MODE_I2C

FMPI2C Master or Slave mode

LL_FMPI2C_MODE_SMBUS_HOST

SMBus Host address acknowledge

LL_FMPI2C_MODE_SMBUS_DEVICE

SMBus Device default mode (Default address not acknowledge)

LL_FMPI2C_MODE_SMBUS_DEVICE_ARP

SMBus Device Default address acknowledge

Transfer Request Direction

LL_FMPI2C_REQUEST_WRITE

Master request a write transfer.

LL_FMPI2C_REQUEST_READ

Master request a read transfer.

SMBus TimeoutA Mode SCL SDA Timeout

LL_FMPI2C_SMBUS_TIMEOUTA_MODE_SCL_LOW

TimeoutA is used to detect SCL low level timeout.

LL_FMPI2C_SMBUS_TIMEOUTA_MODE_SDA_SCL_HIGH

TimeoutA is used to detect both SCL and SDA high level timeout.

SMBus Timeout Selection

LL_FMPI2C_SMBUS_TIMEOUTA

TimeoutA enable bit

LL_FMPI2C_SMBUS_TIMEOUTB

TimeoutB (extended clock) enable bit

LL_FMPI2C_SMBUS_ALL_TIMEOUT

TimeoutA and TimeoutB (extended clock) enable bits

Convert SDA SCL timings

__LL_FMPI2C_CONVERT_TIMINGS

Description:

- Configure the SDA setup, hold time and the SCL high, low period.

Parameters:

- __PRESCALER__: This parameter must be a value between Min_Data=0 and Max_Data=0xF.
- __DATA_SETUP_TIME__: This parameter must be a value between Min_Data=0 and Max_Data=0xF. (tscldel = (SCLDEL+1)xtpresc)
- __DATA_HOLD_TIME__: This parameter must be a value between Min_Data=0 and Max_Data=0xF. (tsdadel = SDADELxtpresc)
- __CLOCK_HIGH_PERIOD__: This parameter must be a value between Min_Data=0 and Max_Data=0xFF. (tsclh = (SCLH+1)xtpresc)
- __CLOCK_LOW_PERIOD__: This parameter must be a value between Min_Data=0 and Max_Data=0xFF. (tscll = (SCLL+1)xtpresc)

Return value:

- Value: between Min_Data=0 and Max_Data=0xFFFFFFFF

Common Write and read registers Macros

LL_FMPI2C_WriteReg

Description:

- Write a value in FMPI2C register.

Parameters:

- __INSTANCE__: FMPI2C Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_FMPI2C_ReadReg

Description:

- Read a value in FMPI2C register.

Parameters:

- __INSTANCE__: FMPI2C Instance
- __REG__: Register to be read

Return value:

- Register: value

81 LL EXTI Generic Driver

81.1 EXTI Firmware driver registers structures

81.1.1 LL_EXTI_InitTypeDef

LL_EXTI_InitTypeDef is defined in the stm32f4xx_ll_exti.h

Data Fields

- ***uint32_t Line_0_31***
- ***FunctionalState LineCommand***
- ***uint8_t Mode***
- ***uint8_t Trigger***

Field Documentation

- ***uint32_t LL_EXTI_InitTypeDef::Line_0_31***
  Specifies the EXTI lines to be enabled or disabled for Lines in range 0 to 31 This parameter can be any combination of ***EXTI_LL_EC_LINE***
- ***FunctionalState LL_EXTI_InitTypeDef::LineCommand***
  Specifies the new state of the selected EXTI lines. This parameter can be set either to ENABLE or DISABLE
- ***uint8_t LL_EXTI_InitTypeDef::Mode***
  Specifies the mode for the EXTI lines. This parameter can be a value of ***EXTI_LL_EC_MODE***.
- ***uint8_t LL_EXTI_InitTypeDef::Trigger***
  Specifies the trigger signal active edge for the EXTI lines. This parameter can be a value of ***EXTI_LL_EC_TRIGGER***.

81.2 EXTI Firmware driver API description

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

81.2.1 Detailed description of functions

LL_EXTI_EnableIT_0_31

Function name

__STATIC_INLINE void LL_EXTI_EnableIT_0_31 (uint32_t ExtiLine)

Function description

Enable ExtiLine Interrupt request for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_17
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22
  - LL_EXTI_LINE_23(*)
  - LL_EXTI_LINE_ALL_0_31

Return values

- **None:**

Notes

- The reset value for the direct or internal lines (see RM) is set to 1 in order to enable the interrupt by default. Bits are set automatically at Power on.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- IMR IMx LL_EXTI_EnableIT_0_31

LL_EXTI_DisableIT_0_31

Function name

__STATIC_INLINE void LL_EXTI_DisableIT_0_31 (uint32_t ExtiLine)

Function description

Disable ExtiLine Interrupt request for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_17
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22
  - LL_EXTI_LINE_23(*)
  - LL_EXTI_LINE_ALL_0_31

Return values

- **None:**

Notes

- The reset value for the direct or internal lines (see RM) is set to 1 in order to enable the interrupt by default. Bits are set automatically at Power on.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- IMR IMx LL_EXTI_DisableIT_0_31

LL_EXTI_IsEnabledIT_0_31

Function name

__STATIC_INLINE uint32_t LL_EXTI_IsEnabledIT_0_31 (uint32_t ExtiLine)

Function description

Indicate if ExtiLine Interrupt request is enabled for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_17
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22
  - LL_EXTI_LINE_23(*)
  - LL_EXTI_LINE_ALL_0_31

Return values

- **State:** of bit (1 or 0).

Notes

- The reset value for the direct or internal lines (see RM) is set to 1 in order to enable the interrupt by default. Bits are set automatically at Power on.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- IMR IMx LL_EXTI_IsEnabledIT_0_31

LL_EXTI_EnableEvent_0_31

Function name

__STATIC_INLINE void LL_EXTI_EnableEvent_0_31 (uint32_t ExtiLine)

Function description

Enable ExtiLine Event request for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_17
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22
  - LL_EXTI_LINE_23(*)
  - LL_EXTI_LINE_ALL_0_31

Return values

- **None:**

Notes

- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- EMR EMx LL_EXTI_EnableEvent_0_31

LL_EXTI_DisableEvent_0_31

Function name

__STATIC_INLINE void LL_EXTI_DisableEvent_0_31 (uint32_t ExtiLine)

Function description

Disable ExtiLine Event request for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_17
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22
  - LL_EXTI_LINE_23(*)
  - LL_EXTI_LINE_ALL_0_31

Return values

- **None:**

Notes

- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- EMR EMx LL_EXTI_DisableEvent_0_31

LL_EXTI_IsEnabledEvent_0_31

Function name

__STATIC_INLINE uint32_t LL_EXTI_IsEnabledEvent_0_31 (uint32_t ExtiLine)

Function description

Indicate if ExtiLine Event request is enabled for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be one of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_17
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22
  - LL_EXTI_LINE_23(*)
  - LL_EXTI_LINE_ALL_0_31

Return values

- **State:** of bit (1 or 0).

Notes

- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- EMR EMx LL_EXTI_IsEnabledEvent_0_31

LL_EXTI_EnableRisingTrig_0_31

Function name

__STATIC_INLINE void LL_EXTI_EnableRisingTrig_0_31 (uint32_t ExtiLine)

Function description

Enable ExtiLine Rising Edge Trigger for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **None:**

Notes

- The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a rising edge on a configurable interrupt line occurs during a write operation in the EXTI_RTSR register, the pending bit is not set. Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- RTSR RTx LL_EXTI_EnableRisingTrig_0_31

LL_EXTI_DisableRisingTrig_0_31

Function name

__STATIC_INLINE void LL_EXTI_DisableRisingTrig_0_31 (uint32_t ExtiLine)

Function description

Disable ExtiLine Rising Edge Trigger for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **None:**

Notes

- The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a rising edge on a configurable interrupt line occurs during a write operation in the EXTI_RTSR register, the pending bit is not set. Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- RTSR RTx LL_EXTI_DisableRisingTrig_0_31

LL_EXTI_IsEnabledRisingTrig_0_31

Function name

__STATIC_INLINE uint32_t LL_EXTI_IsEnabledRisingTrig_0_31 (uint32_t ExtiLine)

Function description

Check if rising edge trigger is enabled for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **State:** of bit (1 or 0).

Notes

- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- RTSR RTx LL_EXTI_IsEnabledRisingTrig_0_31

LL_EXTI_EnableFallingTrig_0_31

Function name

__STATIC_INLINE void LL_EXTI_EnableFallingTrig_0_31 (uint32_t ExtiLine)

Function description

Enable ExtiLine Falling Edge Trigger for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **None:**

Notes

- The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a falling edge on a configurable interrupt line occurs during a write operation in the EXTI_FTSR register, the pending bit is not set. Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- FTSR FTx LL_EXTI_EnableFallingTrig_0_31

LL_EXTI_DisableFallingTrig_0_31

Function name

__STATIC_INLINE void LL_EXTI_DisableFallingTrig_0_31 (uint32_t ExtiLine)

Function description

Disable ExtiLine Falling Edge Trigger for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **None:**

Notes

- The configurable wakeup lines are edge-triggered. No glitch must be generated on these lines. If a Falling edge on a configurable interrupt line occurs during a write operation in the EXTI_FTSR register, the pending bit is not set. Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- FTSR FTx LL_EXTI_DisableFallingTrig_0_31

LL_EXTI_IsEnabledFallingTrig_0_31

Function name

__STATIC_INLINE uint32_t LL_EXTI_IsEnabledFallingTrig_0_31 (uint32_t ExtiLine)

Function description

Check if falling edge trigger is enabled for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **State:** of bit (1 or 0).

Notes

- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- FTSR FTx LL_EXTI_IsEnabledFallingTrig_0_31

LL_EXTI_GenerateSWI_0_31

Function name

__STATIC_INLINE void LL_EXTI_GenerateSWI_0_31 (uint32_t ExtiLine)

Function description

Generate a software Interrupt Event for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **None:**

Notes

- If the interrupt is enabled on this line in the EXTI_IMR, writing a 1 to this bit when it is at '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation. This bit is cleared by clearing the corresponding bit in the EXTI_PR register (by writing a 1 into the bit)
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- SWIER SWIx LL_EXTI_GenerateSWI_0_31

LL_EXTI_IsActiveFlag_0_31

Function name

__STATIC_INLINE uint32_t LL_EXTI_IsActiveFlag_0_31 (uint32_t ExtiLine)

Function description

Check if the ExtLine Flag is set or not for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **State:** of bit (1 or 0).

Notes

- This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a 1 to the bit.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- PR PIFx LL_EXTI_IsActiveFlag_0_31

LL_EXTI_ReadFlag_0_31

Function name

__STATIC_INLINE uint32_t LL_EXTI_ReadFlag_0_31 (uint32_t ExtiLine)

Function description

Read ExtLine Combination Flag for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **@note:** This bit is set when the selected edge event arrives on the interrupt

Notes

- This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a 1 to the bit.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- PR PIFx LL_EXTI_ReadFlag_0_31

LL_EXTI_ClearFlag_0_31

Function name

__STATIC_INLINE void LL_EXTI_ClearFlag_0_31 (uint32_t ExtiLine)

Function description

Clear ExtLine Flags for Lines in range 0 to 31.

Parameters

- **ExtiLine:** This parameter can be a combination of the following values:
  - LL_EXTI_LINE_0
  - LL_EXTI_LINE_1
  - LL_EXTI_LINE_2
  - LL_EXTI_LINE_3
  - LL_EXTI_LINE_4
  - LL_EXTI_LINE_5
  - LL_EXTI_LINE_6
  - LL_EXTI_LINE_7
  - LL_EXTI_LINE_8
  - LL_EXTI_LINE_9
  - LL_EXTI_LINE_10
  - LL_EXTI_LINE_11
  - LL_EXTI_LINE_12
  - LL_EXTI_LINE_13
  - LL_EXTI_LINE_14
  - LL_EXTI_LINE_15
  - LL_EXTI_LINE_16
  - LL_EXTI_LINE_18
  - LL_EXTI_LINE_19(*)
  - LL_EXTI_LINE_20(*)
  - LL_EXTI_LINE_21
  - LL_EXTI_LINE_22

Return values

- **None:**

Notes

- This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a 1 to the bit.
- (*): Available in some devices
- Please check each device line mapping for EXTI Line availability

Reference Manual to LL API cross reference:

- PR PIFx LL_EXTI_ClearFlag_0_31

LL_EXTI_Init

Function name

uint32_t LL_EXTI_Init (LL_EXTI_InitTypeDef * EXTI_InitStruct)

Function description

Initialize the EXTI registers according to the specified parameters in EXTI_InitStruct.

Parameters

- **EXTI_InitStruct:** pointer to a LL_EXTI_InitTypeDef structure.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: EXTI registers are initialized
  - ERROR: not applicable

LL_EXTI_DeInit

Function name

uint32_t LL_EXTI_DeInit (void)

Function description

De-initialize the EXTI registers to their default reset values.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: EXTI registers are de-initialized
  - ERROR: not applicable

LL_EXTI_StructInit

Function name

void LL_EXTI_StructInit (LL_EXTI_InitTypeDef * EXTI_InitStruct)

Function description

Set each LL_EXTI_InitTypeDef field to default value.

Parameters

- **EXTI_InitStruct:** Pointer to a LL_EXTI_InitTypeDef structure.

Return values

- **None:**

81.3 EXTI Firmware driver defines

The following section lists the various define and macros of the module.

81.3.1 EXTI

EXTI

LINE

LL_EXTI_LINE_0

Extended line 0

LL_EXTI_LINE_1

Extended line 1

LL_EXTI_LINE_2

Extended line 2

LL_EXTI_LINE_3

Extended line 3

LL_EXTI_LINE_4

Extended line 4

LL_EXTI_LINE_5

Extended line 5

LL_EXTI_LINE_6

Extended line 6

LL_EXTI_LINE_7

Extended line 7

LL_EXTI_LINE_8

Extended line 8

LL_EXTI_LINE_9

Extended line 9

LL_EXTI_LINE_10

Extended line 10

LL_EXTI_LINE_11

Extended line 11

LL_EXTI_LINE_12

Extended line 12

LL_EXTI_LINE_13

Extended line 13

LL_EXTI_LINE_14

Extended line 14

LL_EXTI_LINE_15

Extended line 15

LL_EXTI_LINE_16

Extended line 16

LL_EXTI_LINE_17

Extended line 17

LL_EXTI_LINE_18

Extended line 18

LL_EXTI_LINE_19

Extended line 19

LL_EXTI_LINE_20

Extended line 20

LL_EXTI_LINE_21

Extended line 21

LL_EXTI_LINE_22

Extended line 22

LL_EXTI_LINE_ALL_0_31

All Extended line not reserved

LL_EXTI_LINE_ALL

All Extended line

LL_EXTI_LINE_NONE

None Extended line

Mode

LL_EXTI_MODE_IT

Interrupt Mode

LL_EXTI_MODE_EVENT

Event Mode

LL_EXTI_MODE_IT_EVENT

Interrupt & Event Mode

Edge Trigger

LL_EXTI_TRIGGER_NONE

No Trigger Mode

LL_EXTI_TRIGGER_RISING

Trigger Rising Mode

LL_EXTI_TRIGGER_FALLING

Trigger Falling Mode

LL_EXTI_TRIGGER_RISING_FALLING

Trigger Rising & Falling Mode

Common Write and read registers Macros

LL_EXTI_WriteReg

Description:

- Write a value in EXTI register.

Parameters:

- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_EXTI_ReadReg

Description:

- Read a value in EXTI register.

Parameters:

- __REG__: Register to be read

Return value:

- Register: value

82 LL GPIO Generic Driver

82.1 GPIO Firmware driver registers structures

82.1.1 LL_GPIO_InitTypeDef

LL_GPIO_InitTypeDef is defined in the stm32f4xx_ll_gpio.h

Data Fields

- ***uint32_t Pin***
- ***uint32_t Mode***
- ***uint32_t Speed***
- ***uint32_t OutputType***
- ***uint32_t Pull***
- ***uint32_t Alternate***

Field Documentation

- ***uint32_t LL_GPIO_InitTypeDef::Pin***

  Specifies the GPIO pins to be configured. This parameter can be any value of ***GPIO_LL_EC_PIN***
- ***uint32_t LL_GPIO_InitTypeDef::Mode***

  Specifies the operating mode for the selected pins. This parameter can be a value of ***GPIO_LL_EC_MODE***.GPIO HW configuration can be modified afterwards using unitary function **LL_GPIO_SetPinMode()**.
- ***uint32_t LL_GPIO_InitTypeDef::Speed***

  Specifies the speed for the selected pins. This parameter can be a value of ***GPIO_LL_EC_SPEED***.GPIO HW configuration can be modified afterwards using unitary function **LL_GPIO_SetPinSpeed()**.
- ***uint32_t LL_GPIO_InitTypeDef::OutputType***

  Specifies the operating output type for the selected pins. This parameter can be a value of ***GPIO_LL_EC_OUTPUT***.GPIO HW configuration can be modified afterwards using unitary function **LL_GPIO_SetPinOutputType()**.
- ***uint32_t LL_GPIO_InitTypeDef::Pull***

  Specifies the operating Pull-up/Pull down for the selected pins. This parameter can be a value of ***GPIO_LL_EC_PULL***.GPIO HW configuration can be modified afterwards using unitary function **LL_GPIO_SetPinPull()**.
- ***uint32_t LL_GPIO_InitTypeDef::Alternate***

  Specifies the Peripheral to be connected to the selected pins. This parameter can be a value of ***GPIO_LL_EC_AF***.GPIO HW configuration can be modified afterwards using unitary function **LL_GPIO_SetAFPin_0_7()** and **LL_GPIO_SetAFPin_8_15()**.

82.2 GPIO Firmware driver API description

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

82.2.1 Detailed description of functions

LL_GPIO_SetPinMode

Function name

__STATIC_INLINE void LL_GPIO_SetPinMode (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Mode)

Function description

Configure gpio mode for a dedicated pin on dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
- **Mode:** This parameter can be one of the following values:
  - LL_GPIO_MODE_INPUT
  - LL_GPIO_MODE_OUTPUT
  - LL_GPIO_MODE_ALTERNATE
  - LL_GPIO_MODE_ANALOG

Return values

- **None:**

Notes

- I/O mode can be Input mode, General purpose output, Alternate function mode or Analog.
- Warning: only one pin can be passed as parameter.

Reference Manual to LL API cross reference:

- MODER MODEy LL_GPIO_SetPinMode

LL_GPIO_GetPinMode

Function name

__STATIC_INLINE uint32_t LL_GPIO_GetPinMode (GPIO_TypeDef * GPIOx, uint32_t Pin)

Function description

Return gpio mode for a dedicated pin on dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15

Return values

- **Returned:** value can be one of the following values:
  - LL_GPIO_MODE_INPUT
  - LL_GPIO_MODE_OUTPUT
  - LL_GPIO_MODE_ALTERNATE
  - LL_GPIO_MODE_ANALOG

Notes

- I/O mode can be Input mode, General purpose output, Alternate function mode or Analog.
- Warning: only one pin can be passed as parameter.

Reference Manual to LL API cross reference:

- MODER MODEy LL_GPIO_GetPinMode

LL_GPIO_SetPinOutputType

Function name

__STATIC_INLINE void LL_GPIO_SetPinOutputType (GPIO_TypeDef * GPIOx, uint32_t PinMask, uint32_t OutputType)

Function description

Configure gpio output type for several pins on dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL
- **OutputType:** This parameter can be one of the following values:
  - LL_GPIO_OUTPUT_PUSHPULL
  - LL_GPIO_OUTPUT_OPENDRAIN

Return values

- **None:**

Notes

- Output type as to be set when gpio pin is in output or alternate modes. Possible type are Push-pull or Open-drain.

Reference Manual to LL API cross reference:

- OTYPER OTy LL_GPIO_SetPinOutputType

LL_GPIO_GetPinOutputType

Function name

__STATIC_INLINE uint32_t LL_GPIO_GetPinOutputType (GPIO_TypeDef * GPIOx, uint32_t Pin)

Function description

Return gpio output type for several pins on dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL

Return values

- **Returned:** value can be one of the following values:
  - LL_GPIO_OUTPUT_PUSHPULL
  - LL_GPIO_OUTPUT_OPENDRAIN

Notes

- Output type as to be set when gpio pin is in output or alternate modes. Possible type are Push-pull or Open-drain.
- Warning: only one pin can be passed as parameter.

Reference Manual to LL API cross reference:

- OTYPER OTy LL_GPIO_GetPinOutputType

LL_GPIO_SetPinSpeed

Function name

__STATIC_INLINE void LL_GPIO_SetPinSpeed (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Speed)

Function description

Configure gpio speed for a dedicated pin on dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
- **Speed:** This parameter can be one of the following values:
  - LL_GPIO_SPEED_FREQ_LOW
  - LL_GPIO_SPEED_FREQ_MEDIUM
  - LL_GPIO_SPEED_FREQ_HIGH
  - LL_GPIO_SPEED_FREQ_VERY_HIGH

Return values

- **None:**

Notes

- I/O speed can be Low, Medium, Fast or High speed.
- Warning: only one pin can be passed as parameter.
- Refer to datasheet for frequency specifications and the power supply and load conditions for each speed.

Reference Manual to LL API cross reference:

- OSPEEDR OSPEEDy LL_GPIO_SetPinSpeed

LL_GPIO_GetPinSpeed

Function name

__STATIC_INLINE uint32_t LL_GPIO_GetPinSpeed (GPIO_TypeDef * GPIOx, uint32_t Pin)

Function description

Return gpio speed for a dedicated pin on dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15

Return values

- **Returned:** value can be one of the following values:
  - LL_GPIO_SPEED_FREQ_LOW
  - LL_GPIO_SPEED_FREQ_MEDIUM
  - LL_GPIO_SPEED_FREQ_HIGH
  - LL_GPIO_SPEED_FREQ_VERY_HIGH

Notes

- I/O speed can be Low, Medium, Fast or High speed.
- Warning: only one pin can be passed as parameter.
- Refer to datasheet for frequency specifications and the power supply and load conditions for each speed.

Reference Manual to LL API cross reference:

- OSPEEDR OSPEEDy LL_GPIO_GetPinSpeed

LL_GPIO_SetPinPull

Function name

__STATIC_INLINE void LL_GPIO_SetPinPull (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Pull)

Function description

Configure gpio pull-up or pull-down for a dedicated pin on a dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
- **Pull:** This parameter can be one of the following values:
  - LL_GPIO_PULL_NO
  - LL_GPIO_PULL_UP
  - LL_GPIO_PULL_DOWN

Return values

- **None:**

Notes

- Warning: only one pin can be passed as parameter.

Reference Manual to LL API cross reference:

- PUPDR PUPDy LL_GPIO_SetPinPull

LL_GPIO_GetPinPull

Function name

__STATIC_INLINE uint32_t LL_GPIO_GetPinPull (GPIO_TypeDef * GPIOx, uint32_t Pin)

Function description

Return gpio pull-up or pull-down for a dedicated pin on a dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15

Return values

- **Returned:** value can be one of the following values:
  - LL_GPIO_PULL_NO
  - LL_GPIO_PULL_UP
  - LL_GPIO_PULL_DOWN

Notes

- Warning: only one pin can be passed as parameter.

Reference Manual to LL API cross reference:

- PUPDR PUPDy LL_GPIO_GetPinPull

LL_GPIO_SetAFPin_0_7

Function name

__STATIC_INLINE void LL_GPIO_SetAFPin_0_7 (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Alternate)

Function description

Configure gpio alternate function of a dedicated pin from 0 to 7 for a dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
- **Alternate:** This parameter can be one of the following values:
  - LL_GPIO_AF_0
  - LL_GPIO_AF_1
  - LL_GPIO_AF_2
  - LL_GPIO_AF_3
  - LL_GPIO_AF_4
  - LL_GPIO_AF_5
  - LL_GPIO_AF_6
  - LL_GPIO_AF_7
  - LL_GPIO_AF_8
  - LL_GPIO_AF_9
  - LL_GPIO_AF_10
  - LL_GPIO_AF_11
  - LL_GPIO_AF_12
  - LL_GPIO_AF_13
  - LL_GPIO_AF_14
  - LL_GPIO_AF_15

Return values

- **None:**

Notes

- Possible values are from AF0 to AF15 depending on target.
- Warning: only one pin can be passed as parameter.

Reference Manual to LL API cross reference:

- AFRL AFSELy LL_GPIO_SetAFPin_0_7

LL_GPIO_GetAFPin_0_7

Function name

__STATIC_INLINE uint32_t LL_GPIO_GetAFPin_0_7 (GPIO_TypeDef * GPIOx, uint32_t Pin)

Function description

Return gpio alternate function of a dedicated pin from 0 to 7 for a dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7

Return values

- **Returned:** value can be one of the following values:
  - LL_GPIO_AF_0
  - LL_GPIO_AF_1
  - LL_GPIO_AF_2
  - LL_GPIO_AF_3
  - LL_GPIO_AF_4
  - LL_GPIO_AF_5
  - LL_GPIO_AF_6
  - LL_GPIO_AF_7
  - LL_GPIO_AF_8
  - LL_GPIO_AF_9
  - LL_GPIO_AF_10
  - LL_GPIO_AF_11
  - LL_GPIO_AF_12
  - LL_GPIO_AF_13
  - LL_GPIO_AF_14
  - LL_GPIO_AF_15

Reference Manual to LL API cross reference:

- AFRL AFSELy LL_GPIO_GetAFPin_0_7

LL_GPIO_SetAFPin_8_15

Function name

__STATIC_INLINE void LL_GPIO_SetAFPin_8_15 (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Alternate)

Function description

Configure gpio alternate function of a dedicated pin from 8 to 15 for a dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
- **Alternate:** This parameter can be one of the following values:
  - LL_GPIO_AF_0
  - LL_GPIO_AF_1
  - LL_GPIO_AF_2
  - LL_GPIO_AF_3
  - LL_GPIO_AF_4
  - LL_GPIO_AF_5
  - LL_GPIO_AF_6
  - LL_GPIO_AF_7
  - LL_GPIO_AF_8
  - LL_GPIO_AF_9
  - LL_GPIO_AF_10
  - LL_GPIO_AF_11
  - LL_GPIO_AF_12
  - LL_GPIO_AF_13
  - LL_GPIO_AF_14
  - LL_GPIO_AF_15

Return values

- **None:**

Notes

- Possible values are from AF0 to AF15 depending on target.
- Warning: only one pin can be passed as parameter.

Reference Manual to LL API cross reference:

- AFRH AFSELy LL_GPIO_SetAFPin_8_15

LL_GPIO_GetAFPin_8_15

Function name

__STATIC_INLINE uint32_t LL_GPIO_GetAFPin_8_15 (GPIO_TypeDef * GPIOx, uint32_t Pin)

Function description

Return gpio alternate function of a dedicated pin from 8 to 15 for a dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **Pin:** This parameter can be one of the following values:
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15

Return values

- **Returned:** value can be one of the following values:
  - LL_GPIO_AF_0
  - LL_GPIO_AF_1
  - LL_GPIO_AF_2
  - LL_GPIO_AF_3
  - LL_GPIO_AF_4
  - LL_GPIO_AF_5
  - LL_GPIO_AF_6
  - LL_GPIO_AF_7
  - LL_GPIO_AF_8
  - LL_GPIO_AF_9
  - LL_GPIO_AF_10
  - LL_GPIO_AF_11
  - LL_GPIO_AF_12
  - LL_GPIO_AF_13
  - LL_GPIO_AF_14
  - LL_GPIO_AF_15

Notes

- Possible values are from AF0 to AF15 depending on target.

Reference Manual to LL API cross reference:

- AFRH AFSELy LL_GPIO_GetAFPin_8_15

LL_GPIO_LockPin

Function name

__STATIC_INLINE void LL_GPIO_LockPin (GPIO_TypeDef * GPIOx, uint32_t PinMask)

Function description

Lock configuration of several pins for a dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL

Return values

- **None:**

Notes

- When the lock sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next reset.
- Each lock bit freezes a specific configuration register (control and alternate function registers).

Reference Manual to LL API cross reference:

- LCKR LCKK LL_GPIO_LockPin

LL_GPIO_IsPinLocked

Function name

__STATIC_INLINE uint32_t LL_GPIO_IsPinLocked (GPIO_TypeDef * GPIOx, uint32_t PinMask)

Function description

Return 1 if all pins passed as parameter, of a dedicated port, are locked.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LCKR LCKy LL_GPIO_IsPinLocked

LL_GPIO_IsAnyPinLocked

Function name

__STATIC_INLINE uint32_t LL_GPIO_IsAnyPinLocked (GPIO_TypeDef * GPIOx)

Function description

Return 1 if one of the pin of a dedicated port is locked.

Parameters

- **GPIOx:** GPIO Port

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- LCKR LCKK LL_GPIO_IsAnyPinLocked

LL_GPIO_ReadInputPort

Function name

__STATIC_INLINE uint32_t LL_GPIO_ReadInputPort (GPIO_TypeDef * GPIOx)

Function description

Return full input data register value for a dedicated port.

Parameters

- **GPIOx:** GPIO Port

Return values

- **Input:** data register value of port

Reference Manual to LL API cross reference:

- IDR IDy LL_GPIO_ReadInputPort

LL_GPIO_IsInputPinSet

Function name

__STATIC_INLINE uint32_t LL_GPIO_IsInputPinSet (GPIO_TypeDef * GPIOx, uint32_t PinMask)

Function description

Return if input data level for several pins of dedicated port is high or low.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- IDR IDy LL_GPIO_IsInputPinSet

LL_GPIO_WriteOutputPort

Function name

__STATIC_INLINE void LL_GPIO_WriteOutputPort (GPIO_TypeDef * GPIOx, uint32_t PortValue)

Function description

Write output data register for the port.

Parameters

- **GPIOx:** GPIO Port
- **PortValue:** Level value for each pin of the port

Return values

- **None:**

Reference Manual to LL API cross reference:

- ODR ODy LL_GPIO_WriteOutputPort

LL_GPIO_ReadOutputPort

Function name

__STATIC_INLINE uint32_t LL_GPIO_ReadOutputPort (GPIO_TypeDef * GPIOx)

Function description

Return full output data register value for a dedicated port.

Parameters

- **GPIOx:** GPIO Port

Return values

- **Output:** data register value of port

Reference Manual to LL API cross reference:

- ODR ODy LL_GPIO_ReadOutputPort

LL_GPIO_IsOutputPinSet

Function name

__STATIC_INLINE uint32_t LL_GPIO_IsOutputPinSet (GPIO_TypeDef * GPIOx, uint32_t PinMask)

Function description

Return if input data level for several pins of dedicated port is high or low.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ODR ODy LL_GPIO_IsOutputPinSet

LL_GPIO_SetOutputPin

Function name

__STATIC_INLINE void LL_GPIO_SetOutputPin (GPIO_TypeDef * GPIOx, uint32_t PinMask)

Function description

Set several pins to high level on dedicated gpio port.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL

Return values

- **None:**

Reference Manual to LL API cross reference:

- BSRR BSy LL_GPIO_SetOutputPin

LL_GPIO_ResetOutputPin

Function name

__STATIC_INLINE void LL_GPIO_ResetOutputPin (GPIO_TypeDef * GPIOx, uint32_t PinMask)

Function description

Set several pins to low level on dedicated gpio port.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL

Return values

- **None:**

Reference Manual to LL API cross reference:

- BSRR BRy LL_GPIO_ResetOutputPin

LL_GPIO_TogglePin

Function name

__STATIC_INLINE void LL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint32_t PinMask)

Function description

Toggle data value for several pin of dedicated port.

Parameters

- **GPIOx:** GPIO Port
- **PinMask:** This parameter can be a combination of the following values:
  - LL_GPIO_PIN_0
  - LL_GPIO_PIN_1
  - LL_GPIO_PIN_2
  - LL_GPIO_PIN_3
  - LL_GPIO_PIN_4
  - LL_GPIO_PIN_5
  - LL_GPIO_PIN_6
  - LL_GPIO_PIN_7
  - LL_GPIO_PIN_8
  - LL_GPIO_PIN_9
  - LL_GPIO_PIN_10
  - LL_GPIO_PIN_11
  - LL_GPIO_PIN_12
  - LL_GPIO_PIN_13
  - LL_GPIO_PIN_14
  - LL_GPIO_PIN_15
  - LL_GPIO_PIN_ALL

Return values

- **None:**

Reference Manual to LL API cross reference:

- ODR ODy LL_GPIO_TogglePin

LL_GPIO_DeInit

Function name

ErrorStatus LL_GPIO_DeInit (GPIO_TypeDef * GPIOx)

Function description

De-initialize GPIO registers (Registers restored to their default values).

Parameters

- **GPIOx:** GPIO Port

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: GPIO registers are de-initialized
  - ERROR: Wrong GPIO Port

LL_GPIO_Init

Function name

ErrorStatus LL_GPIO_Init (GPIO_TypeDef * GPIOx, LL_GPIO_InitTypeDef * GPIO_InitStruct)

Function description

Initialize GPIO registers according to the specified parameters in GPIO_InitStruct.

Parameters

- **GPIOx:** GPIO Port
- **GPIO_InitStruct:** pointer to a LL_GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: GPIO registers are initialized according to GPIO_InitStruct content
  - ERROR: Not applicable

LL_GPIO_StructInit

Function name

void LL_GPIO_StructInit (LL_GPIO_InitTypeDef * GPIO_InitStruct)

Function description

Set each LL_GPIO_InitTypeDef field to default value.

Parameters

- **GPIO_InitStruct:** pointer to a LL_GPIO_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

82.3 GPIO Firmware driver defines

The following section lists the various define and macros of the module.

82.3.1 GPIO

GPIO

Alternate Function

LL_GPIO_AF_0

Select alternate function 0

LL_GPIO_AF_1

Select alternate function 1

LL_GPIO_AF_2

Select alternate function 2

LL_GPIO_AF_3

Select alternate function 3

LL_GPIO_AF_4

Select alternate function 4

LL_GPIO_AF_5

Select alternate function 5

LL_GPIO_AF_6

Select alternate function 6

LL_GPIO_AF_7

Select alternate function 7

LL_GPIO_AF_8

Select alternate function 8

LL_GPIO_AF_9

Select alternate function 9

LL_GPIO_AF_10

Select alternate function 10

LL_GPIO_AF_11

Select alternate function 11

LL_GPIO_AF_12

Select alternate function 12

LL_GPIO_AF_13

Select alternate function 13

LL_GPIO_AF_14

Select alternate function 14

LL_GPIO_AF_15

Select alternate function 15

Mode

LL_GPIO_MODE_INPUT

Select input mode

LL_GPIO_MODE_OUTPUT

Select output mode

LL_GPIO_MODE_ALTERNATE

Select alternate function mode

LL_GPIO_MODE_ANALOG

Select analog mode

Output Type

LL_GPIO_OUTPUT_PUSHPULL

Select push-pull as output type

LL_GPIO_OUTPUT_OPENDRAIN

Select open-drain as output type

PIN

LL_GPIO_PIN_0

Select pin 0

LL_GPIO_PIN_1

Select pin 1

LL_GPIO_PIN_2

Select pin 2

LL_GPIO_PIN_3

Select pin 3

LL_GPIO_PIN_4

Select pin 4

LL_GPIO_PIN_5

Select pin 5

LL_GPIO_PIN_6

Select pin 6

LL_GPIO_PIN_7

Select pin 7

LL_GPIO_PIN_8

Select pin 8

LL_GPIO_PIN_9

Select pin 9

LL_GPIO_PIN_10

Select pin 10

LL_GPIO_PIN_11

Select pin 11

LL_GPIO_PIN_12

Select pin 12

LL_GPIO_PIN_13

Select pin 13

LL_GPIO_PIN_14

Select pin 14

LL_GPIO_PIN_15

Select pin 15

LL_GPIO_PIN_ALL

Select all pins

Pull Up Pull Down

LL_GPIO_PULL_NO

Select I/O no pull

LL_GPIO_PULL_UP

Select I/O pull up

LL_GPIO_PULL_DOWN

Select I/O pull down

Output Speed

LL_GPIO_SPEED_FREQ_LOW

Select I/O low output speed

LL_GPIO_SPEED_FREQ_MEDIUM

Select I/O medium output speed

LL_GPIO_SPEED_FREQ_HIGH

Select I/O fast output speed

LL_GPIO_SPEED_FREQ_VERY_HIGH

Select I/O high output speed

Common Write and read registers Macros

LL_GPIO_WriteReg

Description:

- Write a value in GPIO register.

Parameters:

- __INSTANCE__: GPIO Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_GPIO_ReadReg

Description:

- Read a value in GPIO register.

Parameters:

- __INSTANCE__: GPIO Instance
- __REG__: Register to be read

Return value:

- Register: value

83 LL I2C Generic Driver

83.1 I2C Firmware driver registers structures

83.1.1 LL_I2C_InitTypeDef

LL_I2C_InitTypeDef is defined in the stm32f4xx_ll_i2c.h

Data Fields

- ***uint32_t PeripheralMode***
- ***uint32_t ClockSpeed***
- ***uint32_t DutyCycle***
- ***uint32_t AnalogFilter***
- ***uint32_t DigitalFilter***
- ***uint32_t OwnAddress1***
- ***uint32_t TypeAcknowledge***
- ***uint32_t OwnAddrSize***

Field Documentation

- ***uint32_t LL_I2C_InitTypeDef::PeripheralMode***

  Specifies the peripheral mode. This parameter can be a value of ***I2C_LL_EC_PERIPHERAL_MODE***This feature can be modified afterwards using unitary function **LL_I2C_SetMode()**.

- ***uint32_t LL_I2C_InitTypeDef::ClockSpeed***

  Specifies the clock frequency. This parameter must be set to a value lower than 400kHz (in Hz)This feature can be modified afterwards using unitary function **LL_I2C_SetClockPeriod()** or **LL_I2C_SetDutyCycle()** or **LL_I2C_SetClockSpeedMode()** or **LL_I2C_ConfigSpeed()**.

- ***uint32_t LL_I2C_InitTypeDef::DutyCycle***

  Specifies the I2C fast mode duty cycle. This parameter can be a value of ***I2C_LL_EC_DUTYCYCLE***This feature can be modified afterwards using unitary function **LL_I2C_SetDutyCycle()**.

- ***uint32_t LL_I2C_InitTypeDef::AnalogFilter***

  Enables or disables analog noise filter. This parameter can be a value of ***I2C_LL_EC_ANALOGFILTER_SELECTION***This feature can be modified afterwards using unitary functions **LL_I2C_EnableAnalogFilter()** or **LL_I2C_DisableAnalogFilter()**.

- ***uint32_t LL_I2C_InitTypeDef::DigitalFilter***

  Configures the digital noise filter. This parameter can be a number between Min_Data = 0x00 and Max_Data = 0x0FThis feature can be modified afterwards using unitary function **LL_I2C_SetDigitalFilter()**.

- ***uint32_t LL_I2C_InitTypeDef::OwnAddress1***

  Specifies the device own address 1. This parameter must be a value between Min_Data = 0x00 and Max_Data = 0x3FFThis feature can be modified afterwards using unitary function **LL_I2C_SetOwnAddress1()**.

- ***uint32_t LL_I2C_InitTypeDef::TypeAcknowledge***

  Specifies the ACKnowledge or Non ACKnowledge condition after the address receive match code or next received byte. This parameter can be a value of ***I2C_LL_EC_I2C_ACKNOWLEDGE***This feature can be modified afterwards using unitary function **LL_I2C_AcknowledgeNextData()**.

- ***uint32_t LL_I2C_InitTypeDef::OwnAddrSize***

  Specifies the device own address 1 size (7-bit or 10-bit). This parameter can be a value of ***I2C_LL_EC_OWNADDRESS1***This feature can be modified afterwards using unitary function **LL_I2C_SetOwnAddress1()**.

83.2 I2C Firmware driver API description

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

83.2.1 Detailed description of functions

LL_I2C_Enable

Function name

__STATIC_INLINE void LL_I2C_Enable (I2C_TypeDef * I2Cx)

Function description

Enable I2C peripheral (PE = 1).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 PE LL_I2C_Enable

LL_I2C_Disable

Function name

__STATIC_INLINE void LL_I2C_Disable (I2C_TypeDef * I2Cx)

Function description

Disable I2C peripheral (PE = 0).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 PE LL_I2C_Disable

LL_I2C_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabled (I2C_TypeDef * I2Cx)

Function description

Check if the I2C peripheral is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 PE LL_I2C_IsEnabled

LL_I2C_ConfigFilters

Function name

__STATIC_INLINE void LL_I2C_ConfigFilters (I2C_TypeDef * I2Cx, uint32_t AnalogFilter, uint32_t DigitalFilter)

Function description

Configure Noise Filters (Analog and Digital).

Parameters

- **I2Cx:** I2C Instance.
- **AnalogFilter:** This parameter can be one of the following values:
  - LL_I2C_ANALOGFILTER_ENABLE
  - LL_I2C_ANALOGFILTER_DISABLE
- **DigitalFilter:** This parameter must be a value between Min_Data=0x00 (Digital filter disabled) and Max_Data=0x0F (Digital filter enabled and filtering capability up to 15*TPCLK1) This parameter is used to configure the digital noise filter on SDA and SCL input. The digital filter will suppress the spikes with a length of up to DNF[3:0]*TPCLK1.

Return values

- **None:**

Notes

- If the analog filter is also enabled, the digital filter is added to analog filter. The filters can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- FLTR ANOFF LL_I2C_ConfigFilters
- FLTR DNF LL_I2C_ConfigFilters

LL_I2C_SetDigitalFilter

Function name

__STATIC_INLINE void LL_I2C_SetDigitalFilter (I2C_TypeDef * I2Cx, uint32_t DigitalFilter)

Function description

Configure Digital Noise Filter.

Parameters

- **I2Cx:** I2C Instance.
- **DigitalFilter:** This parameter must be a value between Min_Data=0x00 (Digital filter disabled) and Max_Data=0x0F (Digital filter enabled and filtering capability up to 15*TPCLK1) This parameter is used to configure the digital noise filter on SDA and SCL input. The digital filter will suppress the spikes with a length of up to DNF[3:0]*TPCLK1.

Return values

- **None:**

Notes

- If the analog filter is also enabled, the digital filter is added to analog filter. This filter can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- FLTR DNF LL_I2C_SetDigitalFilter

LL_I2C_GetDigitalFilter

Function name

__STATIC_INLINE uint32_t LL_I2C_GetDigitalFilter (I2C_TypeDef * I2Cx)

Function description

Get the current Digital Noise Filter configuration.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Value:** between Min_Data=0x0 and Max_Data=0xF

Reference Manual to LL API cross reference:

- FLTR DNF LL_I2C_GetDigitalFilter

LL_I2C_EnableAnalogFilter

Function name

__STATIC_INLINE void LL_I2C_EnableAnalogFilter (I2C_TypeDef * I2Cx)

Function description

Enable Analog Noise Filter.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- This filter can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- FLTR ANOFF LL_I2C_EnableAnalogFilter

LL_I2C_DisableAnalogFilter

Function name

__STATIC_INLINE void LL_I2C_DisableAnalogFilter (I2C_TypeDef * I2Cx)

Function description

Disable Analog Noise Filter.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- This filter can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- FLTR ANOFF LL_I2C_DisableAnalogFilter

LL_I2C_IsEnabledAnalogFilter

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledAnalogFilter (I2C_TypeDef * I2Cx)

Function description

Check if Analog Noise Filter is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- FLTR ANOFF LL_I2C_IsEnabledAnalogFilter

LL_I2C_EnableDMAReq_TX

Function name

__STATIC_INLINE void LL_I2C_EnableDMAReq_TX (I2C_TypeDef * I2Cx)

Function description

Enable DMA transmission requests.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 DMAEN LL_I2C_EnableDMAReq_TX

LL_I2C_DisableDMAReq_TX

Function name

__STATIC_INLINE void LL_I2C_DisableDMAReq_TX (I2C_TypeDef * I2Cx)

Function description

Disable DMA transmission requests.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 DMAEN LL_I2C_DisableDMAReq_TX

LL_I2C_IsEnabledDMAReq_TX

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledDMAReq_TX (I2C_TypeDef * I2Cx)

Function description

Check if DMA transmission requests are enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 DMAEN LL_I2C_IsEnabledDMAReq_TX

LL_I2C_EnableDMAReq_RX

Function name

__STATIC_INLINE void LL_I2C_EnableDMAReq_RX (I2C_TypeDef * I2Cx)

Function description

Enable DMA reception requests.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 DMAEN LL_I2C_EnableDMAReq_RX

LL_I2C_DisableDMAReq_RX

Function name

__STATIC_INLINE void LL_I2C_DisableDMAReq_RX (I2C_TypeDef * I2Cx)

Function description

Disable DMA reception requests.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 DMAEN LL_I2C_DisableDMAReq_RX

LL_I2C_IsEnabledDMAReq_RX

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledDMAReq_RX (I2C_TypeDef * I2Cx)

Function description

Check if DMA reception requests are enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 DMAEN LL_I2C_IsEnabledDMAReq_RX

LL_I2C_DMA_GetRegAddr

Function name

__STATIC_INLINE uint32_t LL_I2C_DMA_GetRegAddr (I2C_TypeDef * I2Cx)

Function description

Get the data register address used for DMA transfer.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Address:** of data register

Reference Manual to LL API cross reference:

- DR DR LL_I2C_DMA_GetRegAddr

LL_I2C_EnableClockStretching

Function name

__STATIC_INLINE void LL_I2C_EnableClockStretching (I2C_TypeDef * I2Cx)

Function description

Enable Clock stretching.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- This bit can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR1 NOSTRETCH LL_I2C_EnableClockStretching

LL_I2C_DisableClockStretching

Function name

__STATIC_INLINE void LL_I2C_DisableClockStretching (I2C_TypeDef * I2Cx)

Function description

Disable Clock stretching.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- This bit can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR1 NOSTRETCH LL_I2C_DisableClockStretching

LL_I2C_IsEnabledClockStretching

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledClockStretching (I2C_TypeDef * I2Cx)

Function description

Check if Clock stretching is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 NOSTRETCH LL_I2C_IsEnabledClockStretching

LL_I2C_EnableGeneralCall

Function name

__STATIC_INLINE void LL_I2C_EnableGeneralCall (I2C_TypeDef * I2Cx)

Function description

Enable General Call.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- When enabled the Address 0x00 is ACKed.

Reference Manual to LL API cross reference:

- CR1 ENGC LL_I2C_EnableGeneralCall

LL_I2C_DisableGeneralCall

Function name

__STATIC_INLINE void LL_I2C_DisableGeneralCall (I2C_TypeDef * I2Cx)

Function description

Disable General Call.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- When disabled the Address 0x00 is NACKed.

Reference Manual to LL API cross reference:

- CR1 ENGC LL_I2C_DisableGeneralCall

LL_I2C_IsEnabledGeneralCall

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledGeneralCall (I2C_TypeDef * I2Cx)

Function description

Check if General Call is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 ENGC LL_I2C_IsEnabledGeneralCall

LL_I2C_SetOwnAddress1

Function name

__STATIC_INLINE void LL_I2C_SetOwnAddress1 (I2C_TypeDef * I2Cx, uint32_t OwnAddress1, uint32_t OwnAddrSize)

Function description

Set the Own Address1.

Parameters

- **I2Cx:** I2C Instance.
- **OwnAddress1:** This parameter must be a value between Min_Data=0 and Max_Data=0x3FF.
- **OwnAddrSize:** This parameter can be one of the following values:
  - LL_I2C_OWNADDRESS1_7BIT
  - LL_I2C_OWNADDRESS1_10BIT

Return values

- **None:**

Reference Manual to LL API cross reference:

- OAR1 ADD0 LL_I2C_SetOwnAddress1
- OAR1 ADD1_7 LL_I2C_SetOwnAddress1
- OAR1 ADD8_9 LL_I2C_SetOwnAddress1
- OAR1 ADDMODE LL_I2C_SetOwnAddress1

LL_I2C_SetOwnAddress2

Function name

__STATIC_INLINE void LL_I2C_SetOwnAddress2 (I2C_TypeDef * I2Cx, uint32_t OwnAddress2)

Function description

Set the 7bits Own Address2.

Parameters

- **I2Cx:** I2C Instance.
- **OwnAddress2:** This parameter must be a value between Min_Data=0 and Max_Data=0x7F.

Return values

- **None:**

Notes

- This action has no effect if own address2 is enabled.

Reference Manual to LL API cross reference:

- OAR2 ADD2 LL_I2C_SetOwnAddress2

LL_I2C_EnableOwnAddress2

Function name

__STATIC_INLINE void LL_I2C_EnableOwnAddress2 (I2C_TypeDef * I2Cx)

Function description

Enable acknowledge on Own Address2 match address.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- OAR2 ENDUAL LL_I2C_EnableOwnAddress2

LL_I2C_DisableOwnAddress2

Function name

__STATIC_INLINE void LL_I2C_DisableOwnAddress2 (I2C_TypeDef * I2Cx)

Function description

Disable acknowledge on Own Address2 match address.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- OAR2 ENDUAL LL_I2C_DisableOwnAddress2

LL_I2C_IsEnabledOwnAddress2

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledOwnAddress2 (I2C_TypeDef * I2Cx)

Function description

Check if Own Address1 acknowledge is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- OAR2 ENDUAL LL_I2C_IsEnabledOwnAddress2

LL_I2C_SetPeriphClock

Function name

__STATIC_INLINE void LL_I2C_SetPeriphClock (I2C_TypeDef * I2Cx, uint32_t PeriphClock)

Function description

Configure the Peripheral clock frequency.

Parameters

- **I2Cx:** I2C Instance.
- **PeriphClock:** Peripheral Clock (in Hz)

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 FREQ LL_I2C_SetPeriphClock

LL_I2C_GetPeriphClock

Function name

__STATIC_INLINE uint32_t LL_I2C_GetPeriphClock (I2C_TypeDef * I2Cx)

Function description

Get the Peripheral clock frequency.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Value:** of Peripheral Clock (in Hz)

Reference Manual to LL API cross reference:

- CR2 FREQ LL_I2C_GetPeriphClock

LL_I2C_SetDutyCycle

Function name

__STATIC_INLINE void LL_I2C_SetDutyCycle (I2C_TypeDef * I2Cx, uint32_t DutyCycle)

Function description

Configure the Duty cycle (Fast mode only).

Parameters

- **I2Cx:** I2C Instance.
- **DutyCycle:** This parameter can be one of the following values:
  - LL_I2C_DUTYCYCLE_2
  - LL_I2C_DUTYCYCLE_16_9

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCR DUTY LL_I2C_SetDutyCycle

LL_I2C_GetDutyCycle

Function name

__STATIC_INLINE uint32_t LL_I2C_GetDutyCycle (I2C_TypeDef * I2Cx)

Function description

Get the Duty cycle (Fast mode only).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_I2C_DUTYCYCLE_2
  - LL_I2C_DUTYCYCLE_16_9

Reference Manual to LL API cross reference:

- CCR DUTY LL_I2C_GetDutyCycle

LL_I2C_SetClockSpeedMode

Function name

__STATIC_INLINE void LL_I2C_SetClockSpeedMode (I2C_TypeDef * I2Cx, uint32_t ClockSpeedMode)

Function description

Configure the I2C master clock speed mode.

Parameters

- **I2Cx:** I2C Instance.
- **ClockSpeedMode:** This parameter can be one of the following values:
  - LL_I2C_CLOCK_SPEED_STANDARD_MODE
  - LL_I2C_CLOCK_SPEED_FAST_MODE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCR FS LL_I2C_SetClockSpeedMode

LL_I2C_GetClockSpeedMode

Function name

__STATIC_INLINE uint32_t LL_I2C_GetClockSpeedMode (I2C_TypeDef * I2Cx)

Function description

Get the the I2C master speed mode.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_I2C_CLOCK_SPEED_STANDARD_MODE
  - LL_I2C_CLOCK_SPEED_FAST_MODE

Reference Manual to LL API cross reference:

- CCR FS LL_I2C_GetClockSpeedMode

LL_I2C_SetRiseTime

Function name

__STATIC_INLINE void LL_I2C_SetRiseTime (I2C_TypeDef * I2Cx, uint32_t RiseTime)

Function description

Configure the SCL, SDA rising time.

Parameters

- **I2Cx:** I2C Instance.
- **RiseTime:** This parameter must be a value between Min_Data=0x02 and Max_Data=0x3F.

Return values

- **None:**

Notes

- This bit can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- TRISE TRISE LL_I2C_SetRiseTime

LL_I2C_GetRiseTime

Function name

__STATIC_INLINE uint32_t LL_I2C_GetRiseTime (I2C_TypeDef * I2Cx)

Function description

Get the SCL, SDA rising time.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Value:** between Min_Data=0x02 and Max_Data=0x3F

Reference Manual to LL API cross reference:

- TRISE TRISE LL_I2C_GetRiseTime

LL_I2C_SetClockPeriod

Function name

__STATIC_INLINE void LL_I2C_SetClockPeriod (I2C_TypeDef * I2Cx, uint32_t ClockPeriod)

Function description

Configure the SCL high and low period.

Parameters

- **I2Cx:** I2C Instance.
- **ClockPeriod:** This parameter must be a value between Min_Data=0x004 and Max_Data=0xFFF, except in FAST DUTY mode where Min_Data=0x001.

Return values

- **None:**

Notes

- This bit can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CCR CCR LL_I2C_SetClockPeriod

LL_I2C_GetClockPeriod

Function name

__STATIC_INLINE uint32_t LL_I2C_GetClockPeriod (I2C_TypeDef * I2Cx)

Function description

Get the SCL high and low period.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Value:** between Min_Data=0x004 and Max_Data=0xFFF, except in FAST DUTY mode where Min_Data=0x001.

Reference Manual to LL API cross reference:

- CCR CCR LL_I2C_GetClockPeriod

LL_I2C_ConfigSpeed

Function name

__STATIC_INLINE void LL_I2C_ConfigSpeed (I2C_TypeDef * I2Cx, uint32_t PeriphClock, uint32_t ClockSpeed, uint32_t DutyCycle)

Function description

Configure the SCL speed.

Parameters

- **I2Cx:** I2C Instance.
- **PeriphClock:** Peripheral Clock (in Hz)
- **ClockSpeed:** This parameter must be a value lower than 400kHz (in Hz).
- **DutyCycle:** This parameter can be one of the following values:
  - LL_I2C_DUTYCYCLE_2
  - LL_I2C_DUTYCYCLE_16_9

Return values

- **None:**

Notes

- This bit can only be programmed when the I2C is disabled (PE = 0).

Reference Manual to LL API cross reference:

- CR2 FREQ LL_I2C_ConfigSpeed
- TRISE TRISE LL_I2C_ConfigSpeed
- CCR FS LL_I2C_ConfigSpeed
- CCR DUTY LL_I2C_ConfigSpeed
- CCR CCR LL_I2C_ConfigSpeed

LL_I2C_SetMode

Function name

__STATIC_INLINE void LL_I2C_SetMode (I2C_TypeDef * I2Cx, uint32_t PeripheralMode)

Function description

Configure peripheral mode.

Parameters

- **I2Cx:** I2C Instance.
- **PeripheralMode:** This parameter can be one of the following values:
  - LL_I2C_MODE_I2C
  - LL_I2C_MODE_SMBUS_HOST
  - LL_I2C_MODE_SMBUS_DEVICE
  - LL_I2C_MODE_SMBUS_DEVICE_ARP

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 SMBUS LL_I2C_SetMode
- CR1 SMBTYPE LL_I2C_SetMode
- CR1 ENARP LL_I2C_SetMode

LL_I2C_GetMode

Function name

__STATIC_INLINE uint32_t LL_I2C_GetMode (I2C_TypeDef * I2Cx)

Function description

Get peripheral mode.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_I2C_MODE_I2C
  - LL_I2C_MODE_SMBUS_HOST
  - LL_I2C_MODE_SMBUS_DEVICE
  - LL_I2C_MODE_SMBUS_DEVICE_ARP

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 SMBUS LL_I2C_GetMode
- CR1 SMBTYPE LL_I2C_GetMode
- CR1 ENARP LL_I2C_GetMode

LL_I2C_EnableSMBusAlert

Function name

__STATIC_INLINE void LL_I2C_EnableSMBusAlert (I2C_TypeDef * I2Cx)

Function description

Enable SMBus alert (Host or Device mode)

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
- SMBus Device mode: SMBus Alert pin is drived low and Alert Response Address Header acknowledge is enabled. SMBus Host mode:SMBus Alert pin management is supported.

Reference Manual to LL API cross reference:

- CR1 ALERT LL_I2C_EnableSMBusAlert

LL_I2C_DisableSMBusAlert

Function name

__STATIC_INLINE void LL_I2C_DisableSMBusAlert (I2C_TypeDef * I2Cx)

Function description

Disable SMBus alert (Host or Device mode)

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
- SMBus Device mode: SMBus Alert pin is not drived (can be used as a standard GPIO) and Alert Response Address Header acknowledge is disabled. SMBus Host mode:SMBus Alert pin management is not supported.

Reference Manual to LL API cross reference:

- CR1 ALERT LL_I2C_DisableSMBusAlert

LL_I2C_IsEnabledSMBusAlert

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledSMBusAlert (I2C_TypeDef * I2Cx)

Function description

Check if SMBus alert (Host or Device mode) is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 ALERT LL_I2C_IsEnabledSMBusAlert

LL_I2C_EnableSMBusPEC

Function name

__STATIC_INLINE void LL_I2C_EnableSMBusPEC (I2C_TypeDef * I2Cx)

Function description

Enable SMBus Packet Error Calculation (PEC).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 ENPEC LL_I2C_EnableSMBusPEC

LL_I2C_DisableSMBusPEC

Function name

__STATIC_INLINE void LL_I2C_DisableSMBusPEC (I2C_TypeDef * I2Cx)

Function description

Disable SMBus Packet Error Calculation (PEC).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 ENPEC LL_I2C_DisableSMBusPEC

LL_I2C_IsEnabledSMBusPEC

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledSMBusPEC (I2C_TypeDef * I2Cx)

Function description

Check if SMBus Packet Error Calculation (PEC) is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 ENPEC LL_I2C_IsEnabledSMBusPEC

LL_I2C_EnableIT_TX

Function name

__STATIC_INLINE void LL_I2C_EnableIT_TX (I2C_TypeDef * I2Cx)

Function description

Enable TXE interrupt.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_EnableIT_TX
- CR2 ITBUFEN LL_I2C_EnableIT_TX

LL_I2C_DisableIT_TX

Function name

__STATIC_INLINE void LL_I2C_DisableIT_TX (I2C_TypeDef * I2Cx)

Function description

Disable TXE interrupt.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_DisableIT_TX
- CR2 ITBUFEN LL_I2C_DisableIT_TX

LL_I2C_IsEnabledIT_TX

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_TX (I2C_TypeDef * I2Cx)

Function description

Check if the TXE Interrupt is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_IsEnabledIT_TX
- CR2 ITBUFEN LL_I2C_IsEnabledIT_TX

LL_I2C_EnableIT_RX

Function name

__STATIC_INLINE void LL_I2C_EnableIT_RX (I2C_TypeDef * I2Cx)

Function description

Enable RXNE interrupt.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_EnableIT_RX
- CR2 ITBUFEN LL_I2C_EnableIT_RX

LL_I2C_DisableIT_RX

Function name

__STATIC_INLINE void LL_I2C_DisableIT_RX (I2C_TypeDef * I2Cx)

Function description

Disable RXNE interrupt.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_DisableIT_RX
- CR2 ITBUFEN LL_I2C_DisableIT_RX

LL_I2C_IsEnabledIT_RX

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_RX (I2C_TypeDef * I2Cx)

Function description

Check if the RXNE Interrupt is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_IsEnabledIT_RX
- CR2 ITBUFEN LL_I2C_IsEnabledIT_RX

LL_I2C_EnableIT_EVT

Function name

__STATIC_INLINE void LL_I2C_EnableIT_EVT (I2C_TypeDef * I2Cx)

Function description

Enable Events interrupts.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Any of these events will generate interrupt : Start Bit (SB) Address sent, Address matched (ADDR) 10-bit header sent (ADD10) Stop detection (STOPF) Byte transfer finished (BTF)
- Any of these events will generate interrupt if Buffer interrupts are enabled too(using unitary function LL_I2C_EnableIT_BUF()) : Receive buffer not empty (RXNE) Transmit buffer empty (TXE)

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_EnableIT_EVT

LL_I2C_DisableIT_EVT

Function name

__STATIC_INLINE void LL_I2C_DisableIT_EVT (I2C_TypeDef * I2Cx)

Function description

Disable Events interrupts.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Any of these events will generate interrupt : Start Bit (SB) Address sent, Address matched (ADDR) 10-bit header sent (ADD10) Stop detection (STOPF) Byte transfer finished (BTF) Receive buffer not empty (RXNE) Transmit buffer empty (TXE)

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_DisableIT_EVT

LL_I2C_IsEnabledIT_EVT

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_EVT (I2C_TypeDef * I2Cx)

Function description

Check if Events interrupts are enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 ITEVTEN LL_I2C_IsEnabledIT_EVT

LL_I2C_EnableIT_BUF

Function name

__STATIC_INLINE void LL_I2C_EnableIT_BUF (I2C_TypeDef * I2Cx)

Function description

Enable Buffer interrupts.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Any of these Buffer events will generate interrupt if Events interrupts are enabled too(using unitary function LL_I2C_EnableIT_EVT()) : Receive buffer not empty (RXNE) Transmit buffer empty (TXE)

Reference Manual to LL API cross reference:

- CR2 ITBUFEN LL_I2C_EnableIT_BUF

LL_I2C_DisableIT_BUF

Function name

__STATIC_INLINE void LL_I2C_DisableIT_BUF (I2C_TypeDef * I2Cx)

Function description

Disable Buffer interrupts.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Any of these Buffer events will generate interrupt : Receive buffer not empty (RXNE) Transmit buffer empty (TXE)

Reference Manual to LL API cross reference:

- CR2 ITBUFEN LL_I2C_DisableIT_BUF

LL_I2C_IsEnabledIT_BUF

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_BUF (I2C_TypeDef * I2Cx)

Function description

Check if Buffer interrupts are enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 ITBUFEN LL_I2C_IsEnabledIT_BUF

LL_I2C_EnableIT_ERR

Function name

__STATIC_INLINE void LL_I2C_EnableIT_ERR (I2C_TypeDef * I2Cx)

Function description

Enable Error interrupts.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
- Any of these errors will generate interrupt : Bus Error detection (BERR) Arbitration Loss (ARLO) Acknowledge Failure(AF) Overrun/Underrun (OVR) SMBus Timeout detection (TIMEOUT) SMBus PEC error detection (PECERR) SMBus Alert pin event detection (SMBALERT)

Reference Manual to LL API cross reference:

- CR2 ITERREN LL_I2C_EnableIT_ERR

LL_I2C_DisableIT_ERR

Function name

__STATIC_INLINE void LL_I2C_DisableIT_ERR (I2C_TypeDef * I2Cx)

Function description

Disable Error interrupts.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
- Any of these errors will generate interrupt : Bus Error detection (BERR) Arbitration Loss (ARLO) Acknowledge Failure(AF) Overrun/Underrun (OVR) SMBus Timeout detection (TIMEOUT) SMBus PEC error detection (PECERR) SMBus Alert pin event detection (SMBALERT)

Reference Manual to LL API cross reference:

- CR2 ITERREN LL_I2C_DisableIT_ERR

LL_I2C_IsEnabledIT_ERR

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledIT_ERR (I2C_TypeDef * I2Cx)

Function description

Check if Error interrupts are enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 ITERREN LL_I2C_IsEnabledIT_ERR

LL_I2C_IsActiveFlag_TXE

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_TXE (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Transmit data register empty flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: When next data is written in Transmit data register. SET: When Transmit data register is empty.

Reference Manual to LL API cross reference:

- SR1 TXE LL_I2C_IsActiveFlag_TXE

LL_I2C_IsActiveFlag_BTF

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_BTF (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Byte Transfer Finished flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR1 BTF LL_I2C_IsActiveFlag_BTF

LL_I2C_IsActiveFlag_RXNE

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_RXNE (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Receive data register not empty flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: When Receive data register is read. SET: When the received data is copied in Receive data register.

Reference Manual to LL API cross reference:

- SR1 RXNE LL_I2C_IsActiveFlag_RXNE

LL_I2C_IsActiveFlag_SB

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_SB (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Start Bit (master mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: When No Start condition. SET: When Start condition is generated.

Reference Manual to LL API cross reference:

- SR1 SB LL_I2C_IsActiveFlag_SB

LL_I2C_IsActiveFlag_ADDR

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_ADDR (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Address sent (master mode) or Address matched flag (slave mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When the address is fully sent (master mode) or when the received slave address matched with one of the enabled slave address (slave mode).

Reference Manual to LL API cross reference:

- SR1 ADDR LL_I2C_IsActiveFlag_ADDR

LL_I2C_IsActiveFlag_ADD10

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_ADD10 (I2C_TypeDef * I2Cx)

Function description

Indicate the status of 10-bit header sent (master mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: When no ADD10 event occurred. SET: When the master has sent the first address byte (header).

Reference Manual to LL API cross reference:

- SR1 ADD10 LL_I2C_IsActiveFlag_ADD10

LL_I2C_IsActiveFlag_AF

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_AF (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Acknowledge failure flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: No acknowledge failure. SET: When an acknowledge failure is received after a byte transmission.

Reference Manual to LL API cross reference:

- SR1 AF LL_I2C_IsActiveFlag_AF

LL_I2C_IsActiveFlag_STOP

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_STOP (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Stop detection flag (slave mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When a Stop condition is detected.

Reference Manual to LL API cross reference:

- SR1 STOPF LL_I2C_IsActiveFlag_STOP

LL_I2C_IsActiveFlag_BERR

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_BERR (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Bus error flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When a misplaced Start or Stop condition is detected.

Reference Manual to LL API cross reference:

- SR1 BERR LL_I2C_IsActiveFlag_BERR

LL_I2C_IsActiveFlag_ARLO

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_ARLO (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Arbitration lost flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When arbitration lost.

Reference Manual to LL API cross reference:

- SR1 ARLO LL_I2C_IsActiveFlag_ARLO

LL_I2C_IsActiveFlag_OVR

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_OVR (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Overrun/Underrun flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When an overrun/underrun error occurs (Clock Stretching Disabled).

Reference Manual to LL API cross reference:

- SR1 OVR LL_I2C_IsActiveFlag_OVR

LL_I2C_IsActiveSMBusFlag_PECERR

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_PECERR (I2C_TypeDef * I2Cx)

Function description

Indicate the status of SMBus PEC error flag in reception.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- SR1 PECERR LL_I2C_IsActiveSMBusFlag_PECERR

LL_I2C_IsActiveSMBusFlag_TIMEOUT

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_TIMEOUT (I2C_TypeDef * I2Cx)

Function description

Indicate the status of SMBus Timeout detection flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- SR1 TIMEOUT LL_I2C_IsActiveSMBusFlag_TIMEOUT

LL_I2C_IsActiveSMBusFlag_ALERT

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_ALERT (I2C_TypeDef * I2Cx)

Function description

Indicate the status of SMBus alert flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- SR1 SMBALERT LL_I2C_IsActiveSMBusFlag_ALERT

LL_I2C_IsActiveFlag_BUSY

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_BUSY (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Bus Busy flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Clear default value. SET: When a Start condition is detected.

Reference Manual to LL API cross reference:

- SR2 BUSY LL_I2C_IsActiveFlag_BUSY

LL_I2C_IsActiveFlag_DUAL

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_DUAL (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Dual flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Received address matched with OAR1. SET: Received address matched with OAR2.

Reference Manual to LL API cross reference:

- SR2 DUALF LL_I2C_IsActiveFlag_DUAL

LL_I2C_IsActiveSMBusFlag_SMBHOST

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_SMBHOST (I2C_TypeDef * I2Cx)

Function description

Indicate the status of SMBus Host address reception (Slave mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
- RESET: No SMBus Host address SET: SMBus Host address received.
- This status is cleared by hardware after a STOP condition or repeated START condition.

Reference Manual to LL API cross reference:

- SR2 SMBHOST LL_I2C_IsActiveSMBusFlag_SMBHOST

LL_I2C_IsActiveSMBusFlag_SMBDEFAULT

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveSMBusFlag_SMBDEFAULT (I2C_TypeDef * I2Cx)

Function description

Indicate the status of SMBus Device default address reception (Slave mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
- RESET: No SMBus Device default address SET: SMBus Device default address received.
- This status is cleared by hardware after a STOP condition or repeated START condition.

Reference Manual to LL API cross reference:

- SR2 SMBDEFAULT LL_I2C_IsActiveSMBusFlag_SMBDEFAULT

LL_I2C_IsActiveFlag_GENCALL

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_GENCALL (I2C_TypeDef * I2Cx)

Function description

Indicate the status of General call address reception (Slave mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: No Generall call address SET: General call address received.
- This status is cleared by hardware after a STOP condition or repeated START condition.

Reference Manual to LL API cross reference:

- SR2 GENCALL LL_I2C_IsActiveFlag_GENCALL

LL_I2C_IsActiveFlag_MSL

Function name

__STATIC_INLINE uint32_t LL_I2C_IsActiveFlag_MSL (I2C_TypeDef * I2Cx)

Function description

Indicate the status of Master/Slave flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- RESET: Slave Mode. SET: Master Mode.

Reference Manual to LL API cross reference:

- SR2 MSL LL_I2C_IsActiveFlag_MSL

LL_I2C_ClearFlag_ADDR

Function name

__STATIC_INLINE void LL_I2C_ClearFlag_ADDR (I2C_TypeDef * I2Cx)

Function description

Clear Address Matched flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the I2Cx_SR1 register followed by a read access to the I2Cx_SR2 register.

Reference Manual to LL API cross reference:

- SR1 ADDR LL_I2C_ClearFlag_ADDR

LL_I2C_ClearFlag_AF

Function name

__STATIC_INLINE void LL_I2C_ClearFlag_AF (I2C_TypeDef * I2Cx)

Function description

Clear Acknowledge failure flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR1 AF LL_I2C_ClearFlag_AF

LL_I2C_ClearFlag_STOP

Function name

__STATIC_INLINE void LL_I2C_ClearFlag_STOP (I2C_TypeDef * I2Cx)

Function description

Clear Stop detection flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the I2Cx_SR1 register followed by a write access to I2Cx_CR1 register.

Reference Manual to LL API cross reference:

- SR1 STOPF LL_I2C_ClearFlag_STOP
- CR1 PE LL_I2C_ClearFlag_STOP

LL_I2C_ClearFlag_BERR

Function name

__STATIC_INLINE void LL_I2C_ClearFlag_BERR (I2C_TypeDef * I2Cx)

Function description

Clear Bus error flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR1 BERR LL_I2C_ClearFlag_BERR

LL_I2C_ClearFlag_ARLO

Function name

__STATIC_INLINE void LL_I2C_ClearFlag_ARLO (I2C_TypeDef * I2Cx)

Function description

Clear Arbitration lost flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR1 ARLO LL_I2C_ClearFlag_ARLO

LL_I2C_ClearFlag_OVR

Function name

__STATIC_INLINE void LL_I2C_ClearFlag_OVR (I2C_TypeDef * I2Cx)

Function description

Clear Overrun/Underrun flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR1 OVR LL_I2C_ClearFlag_OVR

LL_I2C_ClearSMBusFlag_PECERR

Function name

__STATIC_INLINE void LL_I2C_ClearSMBusFlag_PECERR (I2C_TypeDef * I2Cx)

Function description

Clear SMBus PEC error flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR1 PECERR LL_I2C_ClearSMBusFlag_PECERR

LL_I2C_ClearSMBusFlag_TIMEOUT

Function name

__STATIC_INLINE void LL_I2C_ClearSMBusFlag_TIMEOUT (I2C_TypeDef * I2Cx)

Function description

Clear SMBus Timeout detection flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- SR1 TIMEOUT LL_I2C_ClearSMBusFlag_TIMEOUT

LL_I2C_ClearSMBusFlag_ALERT

Function name

__STATIC_INLINE void LL_I2C_ClearSMBusFlag_ALERT (I2C_TypeDef * I2Cx)

Function description

Clear SMBus Alert flag.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- SR1 SMBALERT LL_I2C_ClearSMBusFlag_ALERT

LL_I2C_EnableReset

Function name

__STATIC_INLINE void LL_I2C_EnableReset (I2C_TypeDef * I2Cx)

Function description

Enable Reset of I2C peripheral.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 SWRST LL_I2C_EnableReset

LL_I2C_DisableReset

Function name

__STATIC_INLINE void LL_I2C_DisableReset (I2C_TypeDef * I2Cx)

Function description

Disable Reset of I2C peripheral.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 SWRST LL_I2C_DisableReset

LL_I2C_IsResetEnabled

Function name

__STATIC_INLINE uint32_t LL_I2C_IsResetEnabled (I2C_TypeDef * I2Cx)

Function description

Check if the I2C peripheral is under reset state or not.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 SWRST LL_I2C_IsResetEnabled

LL_I2C_AcknowledgeNextData

Function name

__STATIC_INLINE void LL_I2C_AcknowledgeNextData (I2C_TypeDef * I2Cx, uint32_t TypeAcknowledge)

Function description

Prepare the generation of a ACKnowledge or Non ACKnowledge condition after the address receive match code or next received byte.

Parameters

- **I2Cx:** I2C Instance.
- **TypeAcknowledge:** This parameter can be one of the following values:
  - LL_I2C_ACK
  - LL_I2C_NACK

Return values

- **None:**

Notes

- Usage in Slave or Master mode.

Reference Manual to LL API cross reference:

- CR1 ACK LL_I2C_AcknowledgeNextData

LL_I2C_GenerateStartCondition

Function name

__STATIC_INLINE void LL_I2C_GenerateStartCondition (I2C_TypeDef * I2Cx)

Function description

Generate a START or RESTART condition.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- The START bit can be set even if bus is BUSY or I2C is in slave mode. This action has no effect when RELOAD is set.

Reference Manual to LL API cross reference:

- CR1 START LL_I2C_GenerateStartCondition

LL_I2C_GenerateStopCondition

Function name

__STATIC_INLINE void LL_I2C_GenerateStopCondition (I2C_TypeDef * I2Cx)

Function description

Generate a STOP condition after the current byte transfer (master mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 STOP LL_I2C_GenerateStopCondition

LL_I2C_EnableBitPOS

Function name

__STATIC_INLINE void LL_I2C_EnableBitPOS (I2C_TypeDef * I2Cx)

Function description

Enable bit POS (master/host mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- In that case, the ACK bit controls the (N)ACK of the next byte received or the PEC bit indicates that the next byte in shift register is a PEC.

Reference Manual to LL API cross reference:

- CR1 POS LL_I2C_EnableBitPOS

LL_I2C_DisableBitPOS

Function name

__STATIC_INLINE void LL_I2C_DisableBitPOS (I2C_TypeDef * I2Cx)

Function description

Disable bit POS (master/host mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- In that case, the ACK bit controls the (N)ACK of the current byte received or the PEC bit indicates that the current byte in shift register is a PEC.

Reference Manual to LL API cross reference:

- CR1 POS LL_I2C_DisableBitPOS

LL_I2C_IsEnabledBitPOS

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledBitPOS (I2C_TypeDef * I2Cx)

Function description

Check if bit POS is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 POS LL_I2C_IsEnabledBitPOS

LL_I2C_GetTransferDirection

Function name

__STATIC_INLINE uint32_t LL_I2C_GetTransferDirection (I2C_TypeDef * I2Cx)

Function description

Indicate the value of transfer direction.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Returned:** value can be one of the following values:
  - LL_I2C_DIRECTION_WRITE
  - LL_I2C_DIRECTION_READ

Notes

- RESET: Bus is in read transfer (peripheral point of view). SET: Bus is in write transfer (peripheral point of view).

Reference Manual to LL API cross reference:

- SR2 TRA LL_I2C_GetTransferDirection

LL_I2C_EnableLastDMA

Function name

__STATIC_INLINE void LL_I2C_EnableLastDMA (I2C_TypeDef * I2Cx)

Function description

Enable DMA last transfer.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- This action mean that next DMA EOT is the last transfer.

Reference Manual to LL API cross reference:

- CR2 LAST LL_I2C_EnableLastDMA

LL_I2C_DisableLastDMA

Function name

__STATIC_INLINE void LL_I2C_DisableLastDMA (I2C_TypeDef * I2Cx)

Function description

Disable DMA last transfer.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- This action mean that next DMA EOT is not the last transfer.

Reference Manual to LL API cross reference:

- CR2 LAST LL_I2C_DisableLastDMA

LL_I2C_IsEnabledLastDMA

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledLastDMA (I2C_TypeDef * I2Cx)

Function description

Check if DMA last transfer is enabled or disabled.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 LAST LL_I2C_IsEnabledLastDMA

LL_I2C_EnableSMBusPECCompare

Function name

__STATIC_INLINE void LL_I2C_EnableSMBusPECCompare (I2C_TypeDef * I2Cx)

Function description

Enable transfer or internal comparison of the SMBus Packet Error byte (transmission or reception mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.
- This feature is cleared by hardware when the PEC byte is transferred or compared, or by a START or STOP condition, it is also cleared by software.

Reference Manual to LL API cross reference:

- CR1 PEC LL_I2C_EnableSMBusPECCompare

LL_I2C_DisableSMBusPECCompare

Function name

__STATIC_INLINE void LL_I2C_DisableSMBusPECCompare (I2C_TypeDef * I2Cx)

Function description

Disable transfer or internal comparison of the SMBus Packet Error byte (transmission or reception mode).

Parameters

- **I2Cx:** I2C Instance.

Return values

- **None:**

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 PEC LL_I2C_DisableSMBusPECCompare

LL_I2C_IsEnabledSMBusPECCompare

Function name

__STATIC_INLINE uint32_t LL_I2C_IsEnabledSMBusPECCompare (I2C_TypeDef * I2Cx)

Function description

Check if the SMBus Packet Error byte transfer or internal comparison is requested or not.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- CR1 PEC LL_I2C_IsEnabledSMBusPECCompare

LL_I2C_GetSMBusPEC

Function name

__STATIC_INLINE uint32_t LL_I2C_GetSMBusPEC (I2C_TypeDef * I2Cx)

Function description

Get the SMBus Packet Error byte calculated.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Notes

- Macro IS_SMBUS_ALL_INSTANCE(I2Cx) can be used to check whether or not SMBus feature is supported by the I2Cx Instance.

Reference Manual to LL API cross reference:

- SR2 PEC LL_I2C_GetSMBusPEC

LL_I2C_ReceiveData8

Function name

__STATIC_INLINE uint8_t LL_I2C_ReceiveData8 (I2C_TypeDef * I2Cx)

Function description

Read Receive Data register.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **Value:** between Min_Data=0x0 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- DR DR LL_I2C_ReceiveData8

LL_I2C_TransmitData8

Function name

__STATIC_INLINE void LL_I2C_TransmitData8 (I2C_TypeDef * I2Cx, uint8_t Data)

Function description

Write in Transmit Data Register .

Parameters

- **I2Cx:** I2C Instance.
- **Data:** Value between Min_Data=0x0 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR DR LL_I2C_TransmitData8

LL_I2C_Init

Function name

uint32_t LL_I2C_Init (I2C_TypeDef * I2Cx, LL_I2C_InitTypeDef * I2C_InitStruct)

Function description

Initialize the I2C registers according to the specified parameters in I2C_InitStruct.

Parameters

- **I2Cx:** I2C Instance.
- **I2C_InitStruct:** pointer to a LL_I2C_InitTypeDef structure.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS I2C registers are initialized
  - ERROR Not applicable

LL_I2C_DeInit

Function name

uint32_t LL_I2C_DeInit (I2C_TypeDef * I2Cx)

Function description

De-initialize the I2C registers to their default reset values.

Parameters

- **I2Cx:** I2C Instance.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS I2C registers are de-initialized
  - ERROR I2C registers are not de-initialized

LL_I2C_StructInit

Function name

void LL_I2C_StructInit (LL_I2C_InitTypeDef * I2C_InitStruct)

Function description

Set each LL_I2C_InitTypeDef field to default value.

Parameters

- **I2C_InitStruct:** Pointer to a LL_I2C_InitTypeDef structure.

Return values

- **None:**

83.3 I2C Firmware driver defines

The following section lists the various define and macros of the module.

83.3.1 I2C

I2C

Analog Filter Selection

LL_I2C_ANALOGFILTER_ENABLE

Analog filter is enabled.

LL_I2C_ANALOGFILTER_DISABLE

Analog filter is disabled.

Master Clock Speed Mode

LL_I2C_CLOCK_SPEED_STANDARD_MODE

Master clock speed range is standard mode

LL_I2C_CLOCK_SPEED_FAST_MODE

Master clock speed range is fast mode

Read Write Direction

LL_I2C_DIRECTION_WRITE

Bus is in write transfer

LL_I2C_DIRECTION_READ

Bus is in read transfer

Fast Mode Duty Cycle

LL_I2C_DUTYCYCLE_2

I2C fast mode Tlow/Thigh = 2

LL_I2C_DUTYCYCLE_16_9

I2C fast mode Tlow/Thigh = 16/9

Get Flags Defines

LL_I2C_SR1_SB

Start Bit (master mode)

LL_I2C_SR1_ADDR

Address sent (master mode) or Address matched flag (slave mode)

LL_I2C_SR1_BTF

Byte Transfer Finished flag

LL_I2C_SR1_ADD10

10-bit header sent (master mode)

LL_I2C_SR1_STOPF

Stop detection flag (slave mode)

LL_I2C_SR1_RXNE

Data register not empty (receivers)

LL_I2C_SR1_TXE

Data register empty (transmitters)

LL_I2C_SR1_BERR

Bus error

LL_I2C_SR1_ARLO

Arbitration lost

LL_I2C_SR1_AF

Acknowledge failure flag

LL_I2C_SR1_OVR

Overrun/Underrun

LL_I2C_SR1_PECERR

PEC Error in reception (SMBus mode)

LL_I2C_SR1_TIMEOUT

Timeout detection flag (SMBus mode)

LL_I2C_SR1_SMALERT

SMBus alert (SMBus mode)

LL_I2C_SR2_MSL

Master/Slave flag

LL_I2C_SR2_BUSY

Bus busy flag

LL_I2C_SR2_TRA

Transmitter/receiver direction

LL_I2C_SR2_GENCALL

General call address (Slave mode)

LL_I2C_SR2_SMBDEFAULT

SMBus Device default address (Slave mode)

LL_I2C_SR2_SMBHOST

SMBus Host address (Slave mode)

LL_I2C_SR2_DUALF

Dual flag (Slave mode)

Acknowledge Generation

LL_I2C_ACK

ACK is sent after current received byte.

LL_I2C_NACK

NACK is sent after current received byte.

IT Defines

LL_I2C_CR2_ITEVTEN

Events interrupts enable

LL_I2C_CR2_ITBUFEN

Buffer interrupts enable

LL_I2C_CR2_ITERREN

Error interrupts enable

Own Address 1 Length

LL_I2C_OWNADDRESS1_7BIT

Own address 1 is a 7-bit address.

LL_I2C_OWNADDRESS1_10BIT

Own address 1 is a 10-bit address.

Peripheral Mode

LL_I2C_MODE_I2C

I2C Master or Slave mode

LL_I2C_MODE_SMBUS_HOST

SMBus Host address acknowledge

LL_I2C_MODE_SMBUS_DEVICE

SMBus Device default mode (Default address not acknowledge)

LL_I2C_MODE_SMBUS_DEVICE_ARP

SMBus Device Default address acknowledge

Exported_Macros_Helper

__LL_I2C_FREQ_HZ_TO_MHZ

Description:

- Convert Peripheral Clock Frequency in Mhz.

Parameters:

- __PCLK__: This parameter must be a value of peripheral clock (in Hz).

Return value:

- Value: of peripheral clock (in Mhz)

__LL_I2C_FREQ_MHZ_TO_HZ

Description:

- Convert Peripheral Clock Frequency in Hz.

Parameters:

- __PCLK__: This parameter must be a value of peripheral clock (in Mhz).

Return value:

- Value: of peripheral clock (in Hz)

__LL_I2C_RISE_TIME

Description:

- Compute I2C Clock rising time.

Parameters:

- __FREQRANGE__: This parameter must be a value of peripheral clock (in Mhz).
- __SPEED__: This parameter must be a value lower than 400kHz (in Hz).

Return value:

- Value: between Min_Data=0x02 and Max_Data=0x3F

__LL_I2C_SPEED_TO_CCR

Description:

- Compute Speed clock range to a Clock Control Register (I2C_CCR_CCR) value.

Parameters:

- __PCLK__: This parameter must be a value of peripheral clock (in Hz).
- __SPEED__: This parameter must be a value lower than 400kHz (in Hz).
- __DUTYCYCLE__: This parameter can be one of the following values:
  - LL_I2C_DUTYCYCLE_2
  - LL_I2C_DUTYCYCLE_16_9

Return value:

- Value: between Min_Data=0x004 and Max_Data=0xFFF, except in FAST DUTY mode where Min_Data=0x001.

__LL_I2C_SPEED_STANDARD_TO_CCR

Description:

- Compute Speed Standard clock range to a Clock Control Register (I2C_CCR_CCR) value.

Parameters:

- __PCLK__: This parameter must be a value of peripheral clock (in Hz).
- __SPEED__: This parameter must be a value lower than 100kHz (in Hz).

Return value:

- Value: between Min_Data=0x004 and Max_Data=0xFFF.

__LL_I2C_SPEED_FAST_TO_CCR

Description:

- Compute Speed Fast clock range to a Clock Control Register (I2C_CCR_CCR) value.

Parameters:

- __PCLK__: This parameter must be a value of peripheral clock (in Hz).
- __SPEED__: This parameter must be a value between Min_Data=100Khz and Max_Data=400Khz (in Hz).
- __DUTYCYCLE__: This parameter can be one of the following values:
  - LL_I2C_DUTYCYCLE_2
  - LL_I2C_DUTYCYCLE_16_9

Return value:

- Value: between Min_Data=0x001 and Max_Data=0xFFF

__LL_I2C_10BIT_ADDRESS

Description:

- Get the Least significant bits of a 10-Bits address.

Parameters:

- __ADDRESS__: This parameter must be a value of a 10-Bits slave address.

Return value:

- Value: between Min_Data=0x00 and Max_Data=0xFF

__LL_I2C_10BIT_HEADER_WRITE

Description:

- Convert a 10-Bits address to a 10-Bits header with Write direction.

Parameters:

- __ADDRESS__: This parameter must be a value of a 10-Bits slave address.

Return value:

- Value: between Min_Data=0xF0 and Max_Data=0xF6

__LL_I2C_10BIT_HEADER_READ

Description:

- Convert a 10-Bits address to a 10-Bits header with Read direction.

Parameters:

- __ADDRESS__: This parameter must be a value of a 10-Bits slave address.

Return value:

- Value: between Min_Data=0xF1 and Max_Data=0xF7

Common Write and read registers Macros

LL_I2C_WriteReg

Description:

- Write a value in I2C register.

Parameters:

- __INSTANCE__: I2C Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_I2C_ReadReg

Description:

- Read a value in I2C register.

Parameters:

- __INSTANCE__: I2C Instance
- __REG__: Register to be read

Return value:

- Register: value

84 LL IWDG Generic Driver

84.1 IWDG Firmware driver API description

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

84.1.1 Detailed description of functions

LL_IWDG_Enable

Function name

__STATIC_INLINE void LL_IWDG_Enable (IWDG_TypeDef * IWDGx)

Function description

Start the Independent Watchdog.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **None:**

Notes

- Except if the hardware watchdog option is selected

Reference Manual to LL API cross reference:

- KR KEY LL_IWDG_Enable

LL_IWDG_ReloadCounter

Function name

__STATIC_INLINE void LL_IWDG_ReloadCounter (IWDG_TypeDef * IWDGx)

Function description

Reloads IWDG counter with value defined in the reload register.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- KR KEY LL_IWDG_ReloadCounter

LL_IWDG_EnableWriteAccess

Function name

__STATIC_INLINE void LL_IWDG_EnableWriteAccess (IWDG_TypeDef * IWDGx)

Function description

Enable write access to IWDG_PR, IWDG_RLR and IWDG_WINR registers.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- KR KEY LL_IWDG_EnableWriteAccess

LL_IWDG_DisableWriteAccess

Function name

__STATIC_INLINE void LL_IWDG_DisableWriteAccess (IWDG_TypeDef * IWDGx)

Function description

Disable write access to IWDG_PR, IWDG_RLR and IWDG_WINR registers.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- KR KEY LL_IWDG_DisableWriteAccess

LL_IWDG_SetPrescaler

Function name

__STATIC_INLINE void LL_IWDG_SetPrescaler (IWDG_TypeDef * IWDGx, uint32_t Prescaler)

Function description

Select the prescaler of the IWDG.

Parameters

- **IWDGx:** IWDG Instance
- **Prescaler:** This parameter can be one of the following values:
  - LL_IWDG_PRESCALER_4
  - LL_IWDG_PRESCALER_8
  - LL_IWDG_PRESCALER_16
  - LL_IWDG_PRESCALER_32
  - LL_IWDG_PRESCALER_64
  - LL_IWDG_PRESCALER_128
  - LL_IWDG_PRESCALER_256

Return values

- **None:**

Reference Manual to LL API cross reference:

- PR PR LL_IWDG_SetPrescaler

LL_IWDG_GetPrescaler

Function name

__STATIC_INLINE uint32_t LL_IWDG_GetPrescaler (IWDG_TypeDef * IWDGx)

Function description

Get the selected prescaler of the IWDG.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_IWDG_PRESCALER_4
  - LL_IWDG_PRESCALER_8
  - LL_IWDG_PRESCALER_16
  - LL_IWDG_PRESCALER_32
  - LL_IWDG_PRESCALER_64
  - LL_IWDG_PRESCALER_128
  - LL_IWDG_PRESCALER_256

Reference Manual to LL API cross reference:

- PR PR LL_IWDG_GetPrescaler

LL_IWDG_SetReloadCounter

Function name

__STATIC_INLINE void LL_IWDG_SetReloadCounter (IWDG_TypeDef * IWDGx, uint32_t Counter)

Function description

Specify the IWDG down-counter reload value.

Parameters

- **IWDGx:** IWDG Instance
- **Counter:** Value between Min_Data=0 and Max_Data=0x0FFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- RLR RL LL_IWDG_SetReloadCounter

LL_IWDG_GetReloadCounter

Function name

__STATIC_INLINE uint32_t LL_IWDG_GetReloadCounter (IWDG_TypeDef * IWDGx)

Function description

Get the specified IWDG down-counter reload value.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **Value:** between Min_Data=0 and Max_Data=0x0FFF

Reference Manual to LL API cross reference:

- RLR RL LL_IWDG_GetReloadCounter

LL_IWDG_IsActiveFlag_PVU

Function name

__STATIC_INLINE uint32_t LL_IWDG_IsActiveFlag_PVU (IWDG_TypeDef * IWDGx)

Function description

Check if flag Prescaler Value Update is set or not.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR PVU LL_IWDG_IsActiveFlag_PVU

LL_IWDG_IsActiveFlag_RVU

Function name

__STATIC_INLINE uint32_t LL_IWDG_IsActiveFlag_RVU (IWDG_TypeDef * IWDGx)

Function description

Check if flag Reload Value Update is set or not.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR RVU LL_IWDG_IsActiveFlag_RVU

LL_IWDG_IsReady

Function name

__STATIC_INLINE uint32_t LL_IWDG_IsReady (IWDG_TypeDef * IWDGx)

Function description

Check if flags Prescaler & Reload Value Update are reset or not.

Parameters

- **IWDGx:** IWDG Instance

Return values

- **State:** of bits (1 or 0).

Reference Manual to LL API cross reference:

- SR PVU LL_IWDG_IsReady
- SR RVU LL_IWDG_IsReady

84.2 IWDG Firmware driver defines

The following section lists the various define and macros of the module.

84.2.1 IWDG

IWDG

Get Flags Defines

LL_IWDG_SR_PVU

Watchdog prescaler value update

LL_IWDG_SR_RVU

Watchdog counter reload value update

Prescaler Divider

LL_IWDG_PRESCALER_4

Divider by 4

LL_IWDG_PRESCALER_8

Divider by 8

LL_IWDG_PRESCALER_16

Divider by 16

LL_IWDG_PRESCALER_32

Divider by 32

LL_IWDG_PRESCALER_64

Divider by 64

LL_IWDG_PRESCALER_128

Divider by 128

LL_IWDG_PRESCALER_256

Divider by 256

Common Write and read registers Macros

LL_IWDG_WriteReg

Description:

- Write a value in IWDG register.

Parameters:

- __INSTANCE__: IWDG Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_IWDG_ReadReg

Description:

- Read a value in IWDG register.

Parameters:

- __INSTANCE__: IWDG Instance
- __REG__: Register to be read

Return value:

- Register: value

85 LL LPTIM Generic Driver

85.1 LPTIM Firmware driver registers structures

85.1.1 LL_LPTIM_InitTypeDef

LL_LPTIM_InitTypeDef is defined in the stm32f4xx_ll_lptim.h

Data Fields

- ***uint32_t ClockSource***
- ***uint32_t Prescaler***
- ***uint32_t Waveform***
- ***uint32_t Polarity***

Field Documentation

- ***uint32_t LL_LPTIM_InitTypeDef::ClockSource***

  Specifies the source of the clock used by the LPTIM instance. This parameter can be a value of ***LPTIM_LL_EC_CLK_SOURCE***.This feature can be modified afterwards using unitary function **LL_LPTIM_SetClockSource()**.
- ***uint32_t LL_LPTIM_InitTypeDef::Prescaler***

  Specifies the prescaler division ratio. This parameter can be a value of ***LPTIM_LL_EC_PRESCALER***.This feature can be modified afterwards using using unitary function **LL_LPTIM_SetPrescaler()**.
- ***uint32_t LL_LPTIM_InitTypeDef::Waveform***

  Specifies the waveform shape. This parameter can be a value of ***LPTIM_LL_EC_OUTPUT_WAVEFORM***.This feature can be modified afterwards using unitary function **LL_LPTIM_ConfigOutput()**.
- ***uint32_t LL_LPTIM_InitTypeDef::Polarity***

  Specifies waveform polarity. This parameter can be a value of ***LPTIM_LL_EC_OUTPUT_POLARITY***.This feature can be modified afterwards using unitary function **LL_LPTIM_ConfigOutput()**.

85.2 LPTIM Firmware driver API description

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

85.2.1 Detailed description of functions

LL_LPTIM_DeInit

Function name

ErrorStatus LL_LPTIM_DeInit (LPTIM_TypeDef * LPTIMx)

Function description

Set LPTIMx registers to their reset values.

Parameters

- **LPTIMx:** LP Timer instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: LPTIMx registers are de-initialized
  - ERROR: invalid LPTIMx instance

LL_LPTIM_StructInit

Function name

void LL_LPTIM_StructInit (LL_LPTIM_InitTypeDef * LPTIM_InitStruct)

Function description

Set each fields of the LPTIM_InitStruct structure to its default value.

Parameters

- **LPTIM_InitStruct:** pointer to a LL_LPTIM_InitTypeDef structure

Return values

- **None:**

LL_LPTIM_Init

Function name

ErrorStatus LL_LPTIM_Init (LPTIM_TypeDef * LPTIMx, LL_LPTIM_InitTypeDef * LPTIM_InitStruct)

Function description

Configure the LPTIMx peripheral according to the specified parameters.

Parameters

- **LPTIMx:** LP Timer Instance
- **LPTIM_InitStruct:** pointer to a LL_LPTIM_InitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: LPTIMx instance has been initialized
  - ERROR: LPTIMx instance hasn't been initialized

Notes

- LL_LPTIM_Init can only be called when the LPTIM instance is disabled.
- LPTIMx can be disabled using unitary function LL_LPTIM_Disable().

LL_LPTIM_Disable

Function name

void LL_LPTIM_Disable (LPTIM_TypeDef * LPTIMx)

Function description

Disable the LPTIM instance.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Notes

- The following sequence is required to solve LPTIM disable HW limitation. Please check Errata Sheet ES0335 for more details under "MCU may remain stuck in LPTIM interrupt when entering Stop mode" section.

Reference Manual to LL API cross reference:

- CR ENABLE LL_LPTIM_Disable

LL_LPTIM_Enable

Function name

__STATIC_INLINE void LL_LPTIM_Enable (LPTIM_TypeDef * LPTIMx)

Function description

Enable the LPTIM instance.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Notes

- After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM instance is actually enabled.

Reference Manual to LL API cross reference:

- CR ENABLE LL_LPTIM_Enable

LL_LPTIM_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabled (LPTIM_TypeDef * LPTIMx)

Function description

Indicates whether the LPTIM instance is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR ENABLE LL_LPTIM_IsEnabled

LL_LPTIM_StartCounter

Function name

__STATIC_INLINE void LL_LPTIM_StartCounter (LPTIM_TypeDef * LPTIMx, uint32_t OperatingMode)

Function description

Starts the LPTIM counter in the desired mode.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **OperatingMode:** This parameter can be one of the following values:
  - LL_LPTIM_OPERATING_MODE_CONTINUOUS
  - LL_LPTIM_OPERATING_MODE_ONESHOT

Return values

- **None:**

Notes

- LPTIM instance must be enabled before starting the counter.
- It is possible to change on the fly from One Shot mode to Continuous mode.

Reference Manual to LL API cross reference:

- CR CNTSTRT LL_LPTIM_StartCounter
- CR SNGSTRT LL_LPTIM_StartCounter

LL_LPTIM_SetUpdateMode

Function name

__STATIC_INLINE void LL_LPTIM_SetUpdateMode (LPTIM_TypeDef * LPTIMx, uint32_t UpdateMode)

Function description

Set the LPTIM registers update mode (enable/disable register preload)

Parameters

- **LPTIMx:** Low-Power Timer instance
- **UpdateMode:** This parameter can be one of the following values:
  - LL_LPTIM_UPDATE_MODE_IMMEDIATE
  - LL_LPTIM_UPDATE_MODE_ENDOFPERIOD

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.

Reference Manual to LL API cross reference:

- CFGR PRELOAD LL_LPTIM_SetUpdateMode

LL_LPTIM_GetUpdateMode

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetUpdateMode (LPTIM_TypeDef * LPTIMx)

Function description

Get the LPTIM registers update mode.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_UPDATE_MODE_IMMEDIATE
  - LL_LPTIM_UPDATE_MODE_ENDOFPERIOD

Reference Manual to LL API cross reference:

- CFGR PRELOAD LL_LPTIM_GetUpdateMode

LL_LPTIM_SetAutoReload

Function name

__STATIC_INLINE void LL_LPTIM_SetAutoReload (LPTIM_TypeDef * LPTIMx, uint32_t AutoReload)

Function description

Set the auto reload value.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **AutoReload:** Value between Min_Data=0x00 and Max_Data=0xFFFF

Return values

- **None:**

Notes

- The LPTIMx_ARR register content must only be modified when the LPTIM is enabled
- After a write to the LPTIMx_ARR register a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the ARROK flag is set, will lead to unpredictable results.
- autoreload value be strictly greater than the compare value.

Reference Manual to LL API cross reference:

- ARR ARR LL_LPTIM_SetAutoReload

LL_LPTIM_GetAutoReload

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetAutoReload (LPTIM_TypeDef * LPTIMx)

Function description

Get actual auto reload value.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **AutoReload:** Value between Min_Data=0x00 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- ARR ARR LL_LPTIM_GetAutoReload

LL_LPTIM_SetCompare

Function name

__STATIC_INLINE void LL_LPTIM_SetCompare (LPTIM_TypeDef * LPTIMx, uint32_t CompareValue)

Function description

Set the compare value.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **CompareValue:** Value between Min_Data=0x00 and Max_Data=0xFFFF

Return values

- **None:**

Notes

- After a write to the LPTIMx_CMP register a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the CMPOK flag is set, will lead to unpredictable results.

Reference Manual to LL API cross reference:

- CMP CMP LL_LPTIM_SetCompare

LL_LPTIM_GetCompare

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetCompare (LPTIM_TypeDef * LPTIMx)

Function description

Get actual compare value.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **CompareValue:** Value between Min_Data=0x00 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- CMP CMP LL_LPTIM_GetCompare

LL_LPTIM_GetCounter

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetCounter (LPTIM_TypeDef * LPTIMx)

Function description

Get actual counter value.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Counter:** value

Notes

- When the LPTIM instance is running with an asynchronous clock, reading the LPTIMx_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.

Reference Manual to LL API cross reference:

- CNT CNT LL_LPTIM_GetCounter

LL_LPTIM_SetCounterMode

Function name

__STATIC_INLINE void LL_LPTIM_SetCounterMode (LPTIM_TypeDef * LPTIMx, uint32_t CounterMode)

Function description

Set the counter mode (selection of the LPTIM counter clock source).

Parameters

- **LPTIMx:** Low-Power Timer instance
- **CounterMode:** This parameter can be one of the following values:
  - LL_LPTIM_COUNTER_MODE_INTERNAL
  - LL_LPTIM_COUNTER_MODE_EXTERNAL

Return values

- **None:**

Notes

- The counter mode can be set only when the LPTIM instance is disabled.

Reference Manual to LL API cross reference:

- CFGR COUNTMODE LL_LPTIM_SetCounterMode

LL_LPTIM_GetCounterMode

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetCounterMode (LPTIM_TypeDef * LPTIMx)

Function description

Get the counter mode.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_COUNTER_MODE_INTERNAL
  - LL_LPTIM_COUNTER_MODE_EXTERNAL

Reference Manual to LL API cross reference:

- CFGR COUNTMODE LL_LPTIM_GetCounterMode

LL_LPTIM_ConfigOutput

Function name

__STATIC_INLINE void LL_LPTIM_ConfigOutput (LPTIM_TypeDef * LPTIMx, uint32_t Waveform, uint32_t Polarity)

Function description

Configure the LPTIM instance output (LPTIMx_OUT)

Parameters

- **LPTIMx:** Low-Power Timer instance
- **Waveform:** This parameter can be one of the following values:
  - LL_LPTIM_OUTPUT_WAVEFORM_PWM
  - LL_LPTIM_OUTPUT_WAVEFORM_SETONCE
- **Polarity:** This parameter can be one of the following values:
  - LL_LPTIM_OUTPUT_POLARITY_REGULAR
  - LL_LPTIM_OUTPUT_POLARITY_INVERSE

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.
- Regarding the LPTIM output polarity the change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Reference Manual to LL API cross reference:

- CFGR WAVE LL_LPTIM_ConfigOutput
- CFGR WAVPOL LL_LPTIM_ConfigOutput

LL_LPTIM_SetWaveform

Function name

__STATIC_INLINE void LL_LPTIM_SetWaveform (LPTIM_TypeDef * LPTIMx, uint32_t Waveform)

Function description

Set waveform shape.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **Waveform:** This parameter can be one of the following values:
  - LL_LPTIM_OUTPUT_WAVEFORM_PWM
  - LL_LPTIM_OUTPUT_WAVEFORM_SETONCE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR WAVE LL_LPTIM_SetWaveform

LL_LPTIM_GetWaveform

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetWaveform (LPTIM_TypeDef * LPTIMx)

Function description

Get actual waveform shape.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_OUTPUT_WAVEFORM_PWM
  - LL_LPTIM_OUTPUT_WAVEFORM_SETONCE

Reference Manual to LL API cross reference:

- CFGR WAVE LL_LPTIM_GetWaveform

LL_LPTIM_SetPolarity

Function name

__STATIC_INLINE void LL_LPTIM_SetPolarity (LPTIM_TypeDef * LPTIMx, uint32_t Polarity)

Function description

Set output polarity.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **Polarity:** This parameter can be one of the following values:
  - LL_LPTIM_OUTPUT_POLARITY_REGULAR
  - LL_LPTIM_OUTPUT_POLARITY_INVERSE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR WAVPOL LL_LPTIM_SetPolarity

LL_LPTIM_GetPolarity

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetPolarity (LPTIM_TypeDef * LPTIMx)

Function description

Get actual output polarity.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_OUTPUT_POLARITY_REGULAR
  - LL_LPTIM_OUTPUT_POLARITY_INVERSE

Reference Manual to LL API cross reference:

- CFGR WAVPOL LL_LPTIM_GetPolarity

LL_LPTIM_SetPrescaler

Function name

__STATIC_INLINE void LL_LPTIM_SetPrescaler (LPTIM_TypeDef * LPTIMx, uint32_t Prescaler)

Function description

Set actual prescaler division ratio.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **Prescaler:** This parameter can be one of the following values:
  - LL_LPTIM_PRESCALER_DIV1
  - LL_LPTIM_PRESCALER_DIV2
  - LL_LPTIM_PRESCALER_DIV4
  - LL_LPTIM_PRESCALER_DIV8
  - LL_LPTIM_PRESCALER_DIV16
  - LL_LPTIM_PRESCALER_DIV32
  - LL_LPTIM_PRESCALER_DIV64
  - LL_LPTIM_PRESCALER_DIV128

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.
- When the LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated by active edges detected on the LPTIM external Input1, the internal clock provided to the LPTIM must be not be prescaled.

Reference Manual to LL API cross reference:

- CFGR PRESC LL_LPTIM_SetPrescaler

LL_LPTIM_GetPrescaler

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetPrescaler (LPTIM_TypeDef * LPTIMx)

Function description

Get actual prescaler division ratio.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_PRESCALER_DIV1
  - LL_LPTIM_PRESCALER_DIV2
  - LL_LPTIM_PRESCALER_DIV4
  - LL_LPTIM_PRESCALER_DIV8
  - LL_LPTIM_PRESCALER_DIV16
  - LL_LPTIM_PRESCALER_DIV32
  - LL_LPTIM_PRESCALER_DIV64
  - LL_LPTIM_PRESCALER_DIV128

Reference Manual to LL API cross reference:

- CFGR PRESC LL_LPTIM_GetPrescaler

LL_LPTIM_SetInput1Src

Function name

__STATIC_INLINE void LL_LPTIM_SetInput1Src (LPTIM_TypeDef * LPTIMx, uint32_t Src)

Function description

Set LPTIM input 1 source (default GPIO).

Parameters

- **LPTIMx:** Low-Power Timer instance
- **Src:** This parameter can be one of the following values:
  - LL_LPTIM_INPUT1_SRC_PAD_AF
  - LL_LPTIM_INPUT1_SRC_PAD_PA4
  - LL_LPTIM_INPUT1_SRC_PAD_PB9
  - LL_LPTIM_INPUT1_SRC_TIM_DAC

Return values

- **None:**

Reference Manual to LL API cross reference:

- OR OR LL_LPTIM_SetInput1Src

LL_LPTIM_EnableTimeout

Function name

__STATIC_INLINE void LL_LPTIM_EnableTimeout (LPTIM_TypeDef * LPTIMx)

Function description

Enable the timeout function.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.
- The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.
- The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

Reference Manual to LL API cross reference:

- CFGR TIMOUT LL_LPTIM_EnableTimeout

LL_LPTIM_DisableTimeout

Function name

__STATIC_INLINE void LL_LPTIM_DisableTimeout (LPTIM_TypeDef * LPTIMx)

Function description

Disable the timeout function.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.
- A trigger event arriving when the timer is already started will be ignored.

Reference Manual to LL API cross reference:

- CFGR TIMOUT LL_LPTIM_DisableTimeout

LL_LPTIM_IsEnabledTimeout

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledTimeout (LPTIM_TypeDef * LPTIMx)

Function description

Indicate whether the timeout function is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CFGR TIMOUT LL_LPTIM_IsEnabledTimeout

LL_LPTIM_TrigSw

Function name

__STATIC_INLINE void LL_LPTIM_TrigSw (LPTIM_TypeDef * LPTIMx)

Function description

Start the LPTIM counter.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.

Reference Manual to LL API cross reference:

- CFGR TRIGEN LL_LPTIM_TrigSw

LL_LPTIM_ConfigTrigger

Function name

__STATIC_INLINE void LL_LPTIM_ConfigTrigger (LPTIM_TypeDef * LPTIMx, uint32_t Source, uint32_t Filter, uint32_t Polarity)

Function description

Configure the external trigger used as a trigger event for the LPTIM.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **Source:** This parameter can be one of the following values:
  - LL_LPTIM_TRIG_SOURCE_GPIO
  - LL_LPTIM_TRIG_SOURCE_RTCALARMA
  - LL_LPTIM_TRIG_SOURCE_RTCALARMB
  - LL_LPTIM_TRIG_SOURCE_RTCTAMP1
  - LL_LPTIM_TRIG_SOURCE_TIM1_TRGO
  - LL_LPTIM_TRIG_SOURCE_TIM5_TRGO
- **Filter:** This parameter can be one of the following values:
  - LL_LPTIM_TRIG_FILTER_NONE
  - LL_LPTIM_TRIG_FILTER_2
  - LL_LPTIM_TRIG_FILTER_4
  - LL_LPTIM_TRIG_FILTER_8
- **Polarity:** This parameter can be one of the following values:
  - LL_LPTIM_TRIG_POLARITY_RISING
  - LL_LPTIM_TRIG_POLARITY_FALLING
  - LL_LPTIM_TRIG_POLARITY_RISING_FALLING

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.
- An internal clock source must be present when a digital filter is required for the trigger.

Reference Manual to LL API cross reference:

- CFGR TRIGSEL LL_LPTIM_ConfigTrigger
- CFGR TRGFLT LL_LPTIM_ConfigTrigger
- CFGR TRIGEN LL_LPTIM_ConfigTrigger

LL_LPTIM_GetTriggerSource

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetTriggerSource (LPTIM_TypeDef * LPTIMx)

Function description

Get actual external trigger source.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_TRIG_SOURCE_GPIO
  - LL_LPTIM_TRIG_SOURCE_RTCALARMA
  - LL_LPTIM_TRIG_SOURCE_RTCALARMB
  - LL_LPTIM_TRIG_SOURCE_RTCTAMP1
  - LL_LPTIM_TRIG_SOURCE_TIM1_TRGO
  - LL_LPTIM_TRIG_SOURCE_TIM5_TRGO

Reference Manual to LL API cross reference:

- CFGR TRIGSEL LL_LPTIM_GetTriggerSource

LL_LPTIM_GetTriggerFilter

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetTriggerFilter (LPTIM_TypeDef * LPTIMx)

Function description

Get actual external trigger filter.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_TRIG_FILTER_NONE
  - LL_LPTIM_TRIG_FILTER_2
  - LL_LPTIM_TRIG_FILTER_4
  - LL_LPTIM_TRIG_FILTER_8

Reference Manual to LL API cross reference:

- CFGR TRGFLT LL_LPTIM_GetTriggerFilter

LL_LPTIM_GetTriggerPolarity

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetTriggerPolarity (LPTIM_TypeDef * LPTIMx)

Function description

Get actual external trigger polarity.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_TRIG_POLARITY_RISING
  - LL_LPTIM_TRIG_POLARITY_FALLING
  - LL_LPTIM_TRIG_POLARITY_RISING_FALLING

Reference Manual to LL API cross reference:

- CFGR TRIGEN LL_LPTIM_GetTriggerPolarity

LL_LPTIM_SetClockSource

Function name

__STATIC_INLINE void LL_LPTIM_SetClockSource (LPTIM_TypeDef * LPTIMx, uint32_t ClockSource)

Function description

Set the source of the clock used by the LPTIM instance.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **ClockSource:** This parameter can be one of the following values:
  - LL_LPTIM_CLK_SOURCE_INTERNAL
  - LL_LPTIM_CLK_SOURCE_EXTERNAL

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.

Reference Manual to LL API cross reference:

- CFGR CKSEL LL_LPTIM_SetClockSource

LL_LPTIM_GetClockSource

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetClockSource (LPTIM_TypeDef * LPTIMx)

Function description

Get actual LPTIM instance clock source.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_CLK_SOURCE_INTERNAL
  - LL_LPTIM_CLK_SOURCE_EXTERNAL

Reference Manual to LL API cross reference:

- CFGR CKSEL LL_LPTIM_GetClockSource

LL_LPTIM_ConfigClock

Function name

__STATIC_INLINE void LL_LPTIM_ConfigClock (LPTIM_TypeDef * LPTIMx, uint32_t ClockFilter, uint32_t ClockPolarity)

Function description

Configure the active edge or edges used by the counter when the LPTIM is clocked by an external clock source.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **ClockFilter:** This parameter can be one of the following values:
  - LL_LPTIM_CLK_FILTER_NONE
  - LL_LPTIM_CLK_FILTER_2
  - LL_LPTIM_CLK_FILTER_4
  - LL_LPTIM_CLK_FILTER_8
- **ClockPolarity:** This parameter can be one of the following values:
  - LL_LPTIM_CLK_POLARITY_RISING
  - LL_LPTIM_CLK_POLARITY_FALLING
  - LL_LPTIM_CLK_POLARITY_RISING_FALLING

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.
- When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
- An internal clock source must be present when a digital filter is required for external clock.

Reference Manual to LL API cross reference:

- CFGR CKFLT LL_LPTIM_ConfigClock
- CFGR CKPOL LL_LPTIM_ConfigClock

LL_LPTIM_GetClockPolarity

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetClockPolarity (LPTIM_TypeDef * LPTIMx)

Function description

Get actual clock polarity.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_CLK_POLARITY_RISING
  - LL_LPTIM_CLK_POLARITY_FALLING
  - LL_LPTIM_CLK_POLARITY_RISING_FALLING

Reference Manual to LL API cross reference:

- CFGR CKPOL LL_LPTIM_GetClockPolarity

LL_LPTIM_GetClockFilter

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetClockFilter (LPTIM_TypeDef * LPTIMx)

Function description

Get actual clock digital filter.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_CLK_FILTER_NONE
  - LL_LPTIM_CLK_FILTER_2
  - LL_LPTIM_CLK_FILTER_4
  - LL_LPTIM_CLK_FILTER_8

Reference Manual to LL API cross reference:

- CFGR CKFLT LL_LPTIM_GetClockFilter

LL_LPTIM_SetEncoderMode

Function name

__STATIC_INLINE void LL_LPTIM_SetEncoderMode (LPTIM_TypeDef * LPTIMx, uint32_t EncoderMode)

Function description

Configure the encoder mode.

Parameters

- **LPTIMx:** Low-Power Timer instance
- **EncoderMode:** This parameter can be one of the following values:
  - LL_LPTIM_ENCODER_MODE_RISING
  - LL_LPTIM_ENCODER_MODE_FALLING
  - LL_LPTIM_ENCODER_MODE_RISING_FALLING

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.

Reference Manual to LL API cross reference:

- CFGR CKPOL LL_LPTIM_SetEncoderMode

LL_LPTIM_GetEncoderMode

Function name

__STATIC_INLINE uint32_t LL_LPTIM_GetEncoderMode (LPTIM_TypeDef * LPTIMx)

Function description

Get actual encoder mode.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_LPTIM_ENCODER_MODE_RISING
  - LL_LPTIM_ENCODER_MODE_FALLING
  - LL_LPTIM_ENCODER_MODE_RISING_FALLING

Reference Manual to LL API cross reference:

- CFGR CKPOL LL_LPTIM_GetEncoderMode

LL_LPTIM_EnableEncoderMode

Function name

__STATIC_INLINE void LL_LPTIM_EnableEncoderMode (LPTIM_TypeDef * LPTIMx)

Function description

Enable the encoder mode.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.
- In this mode the LPTIM instance must be clocked by an internal clock source. Also, the prescaler division ratio must be equal to 1.
- LPTIM instance must be configured in continuous mode prior enabling the encoder mode.

Reference Manual to LL API cross reference:

- CFGR ENC LL_LPTIM_EnableEncoderMode

LL_LPTIM_DisableEncoderMode

Function name

__STATIC_INLINE void LL_LPTIM_DisableEncoderMode (LPTIM_TypeDef * LPTIMx)

Function description

Disable the encoder mode.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Notes

- This function must be called when the LPTIM instance is disabled.

Reference Manual to LL API cross reference:

- CFGR ENC LL_LPTIM_DisableEncoderMode

LL_LPTIM_IsEnabledEncoderMode

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledEncoderMode (LPTIM_TypeDef * LPTIMx)

Function description

Indicates whether the LPTIM operates in encoder mode.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CFGR ENC LL_LPTIM_IsEnabledEncoderMode

LL_LPTIM_ClearFLAG_CMPM

Function name

__STATIC_INLINE void LL_LPTIM_ClearFLAG_CMPM (LPTIM_TypeDef * LPTIMx)

Function description

Clear the compare match flag (CMPMCF)

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR CMPMCF LL_LPTIM_ClearFLAG_CMPM

LL_LPTIM_IsActiveFlag_CMPM

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsActiveFlag_CMPM (LPTIM_TypeDef * LPTIMx)

Function description

Inform application whether a compare match interrupt has occurred.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR CMPM LL_LPTIM_IsActiveFlag_CMPM

LL_LPTIM_ClearFLAG_ARRM

Function name

__STATIC_INLINE void LL_LPTIM_ClearFLAG_ARRM (LPTIM_TypeDef * LPTIMx)

Function description

Clear the autoreload match flag (ARRMCF)

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR ARRMCF LL_LPTIM_ClearFLAG_ARRM

LL_LPTIM_IsActiveFlag_ARRM

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsActiveFlag_ARRM (LPTIM_TypeDef * LPTIMx)

Function description

Inform application whether a autoreload match interrupt has occurred.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR ARRM LL_LPTIM_IsActiveFlag_ARRM

LL_LPTIM_ClearFlag_EXTTRIG

Function name

__STATIC_INLINE void LL_LPTIM_ClearFlag_EXTTRIG (LPTIM_TypeDef * LPTIMx)

Function description

Clear the external trigger valid edge flag(EXTTRIGCF).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR EXTTRIGCF LL_LPTIM_ClearFlag_EXTTRIG

LL_LPTIM_IsActiveFlag_EXTTRIG

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsActiveFlag_EXTTRIG (LPTIM_TypeDef * LPTIMx)

Function description

Inform application whether a valid edge on the selected external trigger input has occurred.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR EXTTRIG LL_LPTIM_IsActiveFlag_EXTTRIG

LL_LPTIM_ClearFlag_CMPOK

Function name

__STATIC_INLINE void LL_LPTIM_ClearFlag_CMPOK (LPTIM_TypeDef * LPTIMx)

Function description

Clear the compare register update interrupt flag (CMPOKCF).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR CMPOKCF LL_LPTIM_ClearFlag_CMPOK

LL_LPTIM_IsActiveFlag_CMPOK

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsActiveFlag_CMPOK (LPTIM_TypeDef * LPTIMx)

Function description

Informs application whether the APB bus write operation to the LPTIMx_CMP register has been successfully completed.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR CMPOK LL_LPTIM_IsActiveFlag_CMPOK

LL_LPTIM_ClearFlag_ARROK

Function name

__STATIC_INLINE void LL_LPTIM_ClearFlag_ARROK (LPTIM_TypeDef * LPTIMx)

Function description

Clear the autoreload register update interrupt flag (ARROKCF).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR ARROKCF LL_LPTIM_ClearFlag_ARROK

LL_LPTIM_IsActiveFlag_ARROK

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsActiveFlag_ARROK (LPTIM_TypeDef * LPTIMx)

Function description

Informs application whether the APB bus write operation to the LPTIMx_ARR register has been successfully completed.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR ARROK LL_LPTIM_IsActiveFlag_ARROK

LL_LPTIM_ClearFlag_UP

Function name

__STATIC_INLINE void LL_LPTIM_ClearFlag_UP (LPTIM_TypeDef * LPTIMx)

Function description

Clear the counter direction change to up interrupt flag (UPCF).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR UPCF LL_LPTIM_ClearFlag_UP

LL_LPTIM_IsActiveFlag_UP

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsActiveFlag_UP (LPTIM_TypeDef * LPTIMx)

Function description

Informs the application whether the counter direction has changed from down to up (when the LPTIM instance operates in encoder mode).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR UP LL_LPTIM_IsActiveFlag_UP

LL_LPTIM_ClearFlag_DOWN

Function name

__STATIC_INLINE void LL_LPTIM_ClearFlag_DOWN (LPTIM_TypeDef * LPTIMx)

Function description

Clear the counter direction change to down interrupt flag (DOWNCF).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ICR DOWNCF LL_LPTIM_ClearFlag_DOWN

LL_LPTIM_IsActiveFlag_DOWN

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsActiveFlag_DOWN (LPTIM_TypeDef * LPTIMx)

Function description

Informs the application whether the counter direction has changed from up to down (when the LPTIM instance operates in encoder mode).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR DOWN LL_LPTIM_IsActiveFlag_DOWN

LL_LPTIM_EnableIT_CMPM

Function name

__STATIC_INLINE void LL_LPTIM_EnableIT_CMPM (LPTIM_TypeDef * LPTIMx)

Function description

Enable compare match interrupt (CMPMIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER CMPMIE LL_LPTIM_EnableIT_CMPM

LL_LPTIM_DisableIT_CMPM

Function name

__STATIC_INLINE void LL_LPTIM_DisableIT_CMPM (LPTIM_TypeDef * LPTIMx)

Function description

Disable compare match interrupt (CMPMIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER CMPMIE LL_LPTIM_DisableIT_CMPM

LL_LPTIM_IsEnabledIT_CMPM

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledIT_CMPM (LPTIM_TypeDef * LPTIMx)

Function description

Indicates whether the compare match interrupt (CMPMIE) is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- IER CMPMIE LL_LPTIM_IsEnabledIT_CMPM

LL_LPTIM_EnableIT_ARRM

Function name

__STATIC_INLINE void LL_LPTIM_EnableIT_ARRM (LPTIM_TypeDef * LPTIMx)

Function description

Enable autoreload match interrupt (ARRMIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER ARRMIE LL_LPTIM_EnableIT_ARRM

LL_LPTIM_DisableIT_ARRM

Function name

__STATIC_INLINE void LL_LPTIM_DisableIT_ARRM (LPTIM_TypeDef * LPTIMx)

Function description

Disable autoreload match interrupt (ARRMIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER ARRMIE LL_LPTIM_DisableIT_ARRM

LL_LPTIM_IsEnabledIT_ARRM

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledIT_ARRM (LPTIM_TypeDef * LPTIMx)

Function description

Indicates whether the autoreload match interrupt (ARRMIE) is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- IER ARRMIE LL_LPTIM_IsEnabledIT_ARRM

LL_LPTIM_EnableIT_EXTTRIG

Function name

__STATIC_INLINE void LL_LPTIM_EnableIT_EXTTRIG (LPTIM_TypeDef * LPTIMx)

Function description

Enable external trigger valid edge interrupt (EXTTRIGIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER EXTTRIGIE LL_LPTIM_EnableIT_EXTTRIG

LL_LPTIM_DisableIT_EXTTRIG

Function name

__STATIC_INLINE void LL_LPTIM_DisableIT_EXTTRIG (LPTIM_TypeDef * LPTIMx)

Function description

Disable external trigger valid edge interrupt (EXTTRIGIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER EXTTRIGIE LL_LPTIM_DisableIT_EXTTRIG

LL_LPTIM_IsEnabledIT_EXTTRIG

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledIT_EXTTRIG (LPTIM_TypeDef * LPTIMx)

Function description

Indicates external trigger valid edge interrupt (EXTTRIGIE) is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- IER EXTTRIGIE LL_LPTIM_IsEnabledIT_EXTTRIG

LL_LPTIM_EnableIT_CMPOK

Function name

__STATIC_INLINE void LL_LPTIM_EnableIT_CMPOK (LPTIM_TypeDef * LPTIMx)

Function description

Enable compare register write completed interrupt (CMPOKIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER CMPOKIE LL_LPTIM_EnableIT_CMPOK

LL_LPTIM_DisableIT_CMPOK

Function name

__STATIC_INLINE void LL_LPTIM_DisableIT_CMPOK (LPTIM_TypeDef * LPTIMx)

Function description

Disable compare register write completed interrupt (CMPOKIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER CMPOKIE LL_LPTIM_DisableIT_CMPOK

LL_LPTIM_IsEnabledIT_CMPOK

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledIT_CMPOK (LPTIM_TypeDef * LPTIMx)

Function description

Indicates whether the compare register write completed interrupt (CMPOKIE) is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- IER CMPOKIE LL_LPTIM_IsEnabledIT_CMPOK

LL_LPTIM_EnableIT_ARROK

Function name

__STATIC_INLINE void LL_LPTIM_EnableIT_ARROK (LPTIM_TypeDef * LPTIMx)

Function description

Enable autoreload register write completed interrupt (ARROKIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER ARROKIE LL_LPTIM_EnableIT_ARROK

LL_LPTIM_DisableIT_ARROK

Function name

__STATIC_INLINE void LL_LPTIM_DisableIT_ARROK (LPTIM_TypeDef * LPTIMx)

Function description

Disable autoreload register write completed interrupt (ARROKIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER ARROKIE LL_LPTIM_DisableIT_ARROK

LL_LPTIM_IsEnabledIT_ARROK

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledIT_ARROK (LPTIM_TypeDef * LPTIMx)

Function description

Indicates whether the autoreload register write completed interrupt (ARROKIE) is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit(1 or 0).

Reference Manual to LL API cross reference:

- IER ARROKIE LL_LPTIM_IsEnabledIT_ARROK

LL_LPTIM_EnableIT_UP

Function name

__STATIC_INLINE void LL_LPTIM_EnableIT_UP (LPTIM_TypeDef * LPTIMx)

Function description

Enable direction change to up interrupt (UPIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER UPIE LL_LPTIM_EnableIT_UP

LL_LPTIM_DisableIT_UP

Function name

__STATIC_INLINE void LL_LPTIM_DisableIT_UP (LPTIM_TypeDef * LPTIMx)

Function description

Disable direction change to up interrupt (UPIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER UPIE LL_LPTIM_DisableIT_UP

LL_LPTIM_IsEnabledIT_UP

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledIT_UP (LPTIM_TypeDef * LPTIMx)

Function description

Indicates whether the direction change to up interrupt (UPIE) is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit(1 or 0).

Reference Manual to LL API cross reference:

- IER UPIE LL_LPTIM_IsEnabledIT_UP

LL_LPTIM_EnableIT_DOWN

Function name

__STATIC_INLINE void LL_LPTIM_EnableIT_DOWN (LPTIM_TypeDef * LPTIMx)

Function description

Enable direction change to down interrupt (DOWNIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER DOWNIE LL_LPTIM_EnableIT_DOWN

LL_LPTIM_DisableIT_DOWN

Function name

__STATIC_INLINE void LL_LPTIM_DisableIT_DOWN (LPTIM_TypeDef * LPTIMx)

Function description

Disable direction change to down interrupt (DOWNIE).

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- IER DOWNIE LL_LPTIM_DisableIT_DOWN

LL_LPTIM_IsEnabledIT_DOWN

Function name

__STATIC_INLINE uint32_t LL_LPTIM_IsEnabledIT_DOWN (LPTIM_TypeDef * LPTIMx)

Function description

Indicates whether the direction change to down interrupt (DOWNIE) is enabled.

Parameters

- **LPTIMx:** Low-Power Timer instance

Return values

- **State:** of bit(1 or 0).

Reference Manual to LL API cross reference:

- IER DOWNIE LL_LPTIM_IsEnabledIT_DOWN

85.3 LPTIM Firmware driver defines

The following section lists the various define and macros of the module.

85.3.1 LPTIM

LPTIM

Input1 Source

LL_LPTIM_INPUT1_SRC_PAD_AF

LL_LPTIM_INPUT1_SRC_PAD_PA4

LL_LPTIM_INPUT1_SRC_PAD_PB9

LL_LPTIM_INPUT1_SRC_TIM_DAC

Clock Filter

LL_LPTIM_CLK_FILTER_NONE

Any external clock signal level change is considered as a valid transition

LL_LPTIM_CLK_FILTER_2

External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition

LL_LPTIM_CLK_FILTER_4

External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition

LL_LPTIM_CLK_FILTER_8

External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition

Clock Polarity

LL_LPTIM_CLK_POLARITY_RISING

The rising edge is the active edge used for counting

LL_LPTIM_CLK_POLARITY_FALLING

The falling edge is the active edge used for counting

LL_LPTIM_CLK_POLARITY_RISING_FALLING

Both edges are active edges

Clock Source

LL_LPTIM_CLK_SOURCE_INTERNAL

LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

LL_LPTIM_CLK_SOURCE_EXTERNAL

LPTIM is clocked by an external clock source through the LPTIM external Input1

Counter Mode

LL_LPTIM_COUNTER_MODE_INTERNAL

The counter is incremented following each internal clock pulse

LL_LPTIM_COUNTER_MODE_EXTERNAL

The counter is incremented following each valid clock pulse on the LPTIM external Input1

Encoder Mode

LL_LPTIM_ENCODER_MODE_RISING

The rising edge is the active edge used for counting

LL_LPTIM_ENCODER_MODE_FALLING

The falling edge is the active edge used for counting

LL_LPTIM_ENCODER_MODE_RISING_FALLING

Both edges are active edges

Get Flags Defines

LL_LPTIM_ISR_CMPM

Compare match

LL_LPTIM_ISR_ARRM

Autoreload match

LL_LPTIM_ISR_EXTTRIG

External trigger edge event

LL_LPTIM_ISR_CMPOK

Compare register update OK

LL_LPTIM_ISR_ARROK

Autoreload register update OK

LL_LPTIM_ISR_UP

Counter direction change down to up

LL_LPTIM_ISR_DOWN

Counter direction change up to down

IT Defines

LL_LPTIM_IER_CMPMIE

Compare match Interrupt Enable

LL_LPTIM_IER_ARRMIE

Autoreload match Interrupt Enable

LL_LPTIM_IER_EXTTRIGIE

External trigger valid edge Interrupt Enable

LL_LPTIM_IER_CMPOKIE

Compare register update OK Interrupt Enable

LL_LPTIM_IER_ARROKIE

Autoreload register update OK Interrupt Enable

LL_LPTIM_IER_UPIE

Direction change to UP Interrupt Enable

LL_LPTIM_IER_DOWNIE

Direction change to down Interrupt Enable

Operating Mode

LL_LPTIM_OPERATING_MODE_CONTINUOUS

LP Timer starts in continuous mode

LL_LPTIM_OPERATING_MODE_ONESHOT

LP Tilmer starts in single mode

Output Polarity

LL_LPTIM_OUTPUT_POLARITY_REGULAR

The LPTIM output reflects the compare results between LPTIMx_ARR and LPTIMx_CMP registers

LL_LPTIM_OUTPUT_POLARITY_INVERSE

The LPTIM output reflects the inverse of the compare results between LPTIMx_ARR and LPTIMx_CMP registers

Output Waveform Type

LL_LPTIM_OUTPUT_WAVEFORM_PWM

LPTIM generates either a PWM waveform or a One pulse waveform depending on chosen operating mode CONTINOUS or SINGLE

LL_LPTIM_OUTPUT_WAVEFORM_SETONCE

LPTIM generates a Set Once waveform

Prescaler Value

LL_LPTIM_PRESCALER_DIV1

Prescaler division factor is set to 1

LL_LPTIM_PRESCALER_DIV2

Prescaler division factor is set to 2

LL_LPTIM_PRESCALER_DIV4

Prescaler division factor is set to 4

LL_LPTIM_PRESCALER_DIV8

Prescaler division factor is set to 8

LL_LPTIM_PRESCALER_DIV16

Prescaler division factor is set to 16

LL_LPTIM_PRESCALER_DIV32

Prescaler division factor is set to 32

LL_LPTIM_PRESCALER_DIV64

Prescaler division factor is set to 64

LL_LPTIM_PRESCALER_DIV128

Prescaler division factor is set to 128

Trigger Filter

LL_LPTIM_TRIG_FILTER_NONE

Any trigger active level change is considered as a valid trigger

LL_LPTIM_TRIG_FILTER_2

Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger

LL_LPTIM_TRIG_FILTER_4

Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger

LL_LPTIM_TRIG_FILTER_8

Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger

Trigger Polarity

LL_LPTIM_TRIG_POLARITY_RISING

LPTIM counter starts when a rising edge is detected

LL_LPTIM_TRIG_POLARITY_FALLING

LPTIM counter starts when a falling edge is detected

LL_LPTIM_TRIG_POLARITY_RISING_FALLING

LPTIM counter starts when a rising or a falling edge is detected

Trigger Source

LL_LPTIM_TRIG_SOURCE_GPIO

External input trigger is connected to TIMx_ETR input

LL_LPTIM_TRIG_SOURCE_RTCALARMA

External input trigger is connected to RTC Alarm A

LL_LPTIM_TRIG_SOURCE_RTCALARMB

External input trigger is connected to RTC Alarm B

LL_LPTIM_TRIG_SOURCE_RTCTAMP1

External input trigger is connected to RTC Tamper 1

LL_LPTIM_TRIG_SOURCE_TIM1_TRGO

External input trigger is connected to TIM1

LL_LPTIM_TRIG_SOURCE_TIM5_TRGO

External input trigger is connected to TIM5

Update Mode

LL_LPTIM_UPDATE_MODE_IMMEDIATE

Preload is disabled: registers are updated after each APB bus write access

LL_LPTIM_UPDATE_MODE_ENDOFPERIOD

preload is enabled: registers are updated at the end of the current LPTIM period

Common Write and read registers Macros

LL_LPTIM_WriteReg

Description:

- Write a value in LPTIM register.

Parameters:

- __INSTANCE__: LPTIM Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_LPTIM_ReadReg

Description:

- Read a value in LPTIM register.

Parameters:

- __INSTANCE__: LPTIM Instance
- __REG__: Register to be read

Return value:

- Register: value

86 LL PWR Generic Driver

86.1 PWR Firmware driver API description

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

86.1.1 Detailed description of functions

LL_PWR_EnableUnderDriveMode

Function name

__STATIC_INLINE void LL_PWR_EnableUnderDriveMode (void)

Function description

Enable Under Drive Mode.

Return values

- **None:**

Notes

- This mode is enabled only with STOP low power mode. In this mode, the 1.2V domain is preserved in reduced leakage mode. This mode is only available when the main Regulator or the low power Regulator is in low voltage mode.
- If the Under-drive mode was enabled, it is automatically disabled after exiting Stop mode. When the voltage Regulator operates in Under-drive mode, an additional startup delay is induced when waking up from Stop mode.

Reference Manual to LL API cross reference:

- CR UDEN LL_PWR_EnableUnderDriveMode

LL_PWR_DisableUnderDriveMode

Function name

__STATIC_INLINE void LL_PWR_DisableUnderDriveMode (void)

Function description

Disable Under Drive Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR UDEN LL_PWR_DisableUnderDriveMode

LL_PWR_IsEnabledUnderDriveMode

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledUnderDriveMode (void)

Function description

Check if Under Drive Mode is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR UDEN LL_PWR_IsEnabledUnderDriveMode

LL_PWR_EnableOverDriveSwitching

Function name

__STATIC_INLINE void LL_PWR_EnableOverDriveSwitching (void)

Function description

Enable Over drive switching.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR ODSWEN LL_PWR_EnableOverDriveSwitching

LL_PWR_DisableOverDriveSwitching

Function name

__STATIC_INLINE void LL_PWR_DisableOverDriveSwitching (void)

Function description

Disable Over drive switching.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR ODSWEN LL_PWR_DisableOverDriveSwitching

LL_PWR_IsEnabledOverDriveSwitching

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledOverDriveSwitching (void)

Function description

Check if Over drive switching is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR ODSWEN LL_PWR_IsEnabledOverDriveSwitching

LL_PWR_EnableOverDriveMode

Function name

__STATIC_INLINE void LL_PWR_EnableOverDriveMode (void)

Function description

Enable Over drive Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR ODEN LL_PWR_EnableOverDriveMode

LL_PWR_DisableOverDriveMode

Function name

__STATIC_INLINE void LL_PWR_DisableOverDriveMode (void)

Function description

Disable Over drive Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR ODEN LL_PWR_DisableOverDriveMode

LL_PWR_IsEnabledOverDriveMode

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledOverDriveMode (void)

Function description

Check if Over drive switching is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR ODEN LL_PWR_IsEnabledOverDriveMode

LL_PWR_EnableMainRegulatorDeepSleepUDMode

Function name

__STATIC_INLINE void LL_PWR_EnableMainRegulatorDeepSleepUDMode (void)

Function description

Enable Main Regulator in deepsleep under-drive Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR MRUDS LL_PWR_EnableMainRegulatorDeepSleepUDMode

LL_PWR_DisableMainRegulatorDeepSleepUDMode

Function name

__STATIC_INLINE void LL_PWR_DisableMainRegulatorDeepSleepUDMode (void)

Function description

Disable Main Regulator in deepsleep under-drive Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR MRUDS LL_PWR_DisableMainRegulatorDeepSleepUDMode

LL_PWR_IsEnabledMainRegulatorDeepSleepUDMode

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledMainRegulatorDeepSleepUDMode (void)

Function description

Check if Main Regulator in deepsleep under-drive Mode is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR MRUDS LL_PWR_IsEnabledMainRegulatorDeepSleepUDMode

LL_PWR_EnableLowPowerRegulatorDeepSleepUDMode

Function name

__STATIC_INLINE void LL_PWR_EnableLowPowerRegulatorDeepSleepUDMode (void)

Function description

Enable Low Power Regulator in deepsleep under-drive Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR LPUDS LL_PWR_EnableLowPowerRegulatorDeepSleepUDMode

LL_PWR_DisableLowPowerRegulatorDeepSleepUDMode

Function name

__STATIC_INLINE void LL_PWR_DisableLowPowerRegulatorDeepSleepUDMode (void)

Function description

Disable Low Power Regulator in deepsleep under-drive Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR LPUDS LL_PWR_DisableLowPowerRegulatorDeepSleepUDMode

LL_PWR_IsEnabledLowPowerRegulatorDeepSleepUDMode

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledLowPowerRegulatorDeepSleepUDMode (void)

Function description

Check if Low Power Regulator in deepsleep under-drive Mode is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR LPUDS LL_PWR_IsEnabledLowPowerRegulatorDeepSleepUDMode

LL_PWR_EnableMainRegulatorLowVoltageMode

Function name

__STATIC_INLINE void LL_PWR_EnableMainRegulatorLowVoltageMode (void)

Function description

Enable Main Regulator low voltage Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR MRLVDS LL_PWR_EnableMainRegulatorLowVoltageMode

LL_PWR_DisableMainRegulatorLowVoltageMode

Function name

__STATIC_INLINE void LL_PWR_DisableMainRegulatorLowVoltageMode (void)

Function description

Disable Main Regulator low voltage Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR MRLVDS LL_PWR_DisableMainRegulatorLowVoltageMode

LL_PWR_IsEnabledMainRegulatorLowVoltageMode

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledMainRegulatorLowVoltageMode (void)

Function description

Check if Main Regulator low voltage Mode is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR MRLVDS LL_PWR_IsEnabledMainRegulatorLowVoltageMode

LL_PWR_EnableLowPowerRegulatorLowVoltageMode

Function name

__STATIC_INLINE void LL_PWR_EnableLowPowerRegulatorLowVoltageMode (void)

Function description

Enable Low Power Regulator low voltage Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR LPLVDS LL_PWR_EnableLowPowerRegulatorLowVoltageMode

LL_PWR_DisableLowPowerRegulatorLowVoltageMode

Function name

__STATIC_INLINE void LL_PWR_DisableLowPowerRegulatorLowVoltageMode (void)

Function description

Disable Low Power Regulator low voltage Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR LPLVDS LL_PWR_DisableLowPowerRegulatorLowVoltageMode

LL_PWR_IsEnabledLowPowerRegulatorLowVoltageMode

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledLowPowerRegulatorLowVoltageMode (void)

Function description

Check if Low Power Regulator low voltage Mode is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR LPLVDS LL_PWR_IsEnabledLowPowerRegulatorLowVoltageMode

LL_PWR_SetRegulVoltageScaling

Function name

__STATIC_INLINE void LL_PWR_SetRegulVoltageScaling (uint32_t VoltageScaling)

Function description

Set the main internal Regulator output voltage.

Parameters

- **VoltageScaling:** This parameter can be one of the following values:
  - LL_PWR_REGU_VOLTAGE_SCALE1 (*)
  - LL_PWR_REGU_VOLTAGE_SCALE2
  - LL_PWR_REGU_VOLTAGE_SCALE3 (*) LL_PWR_REGU_VOLTAGE_SCALE1 is not available for STM32F401xx devices

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR VOS LL_PWR_SetRegulVoltageScaling

LL_PWR_GetRegulVoltageScaling

Function name

__STATIC_INLINE uint32_t LL_PWR_GetRegulVoltageScaling (void)

Function description

Get the main internal Regulator output voltage.

Return values

- **Returned:** value can be one of the following values:
  - LL_PWR_REGU_VOLTAGE_SCALE1 (*)
  - LL_PWR_REGU_VOLTAGE_SCALE2
  - LL_PWR_REGU_VOLTAGE_SCALE3 (*) LL_PWR_REGU_VOLTAGE_SCALE1 is not available for STM32F401xx devices

Reference Manual to LL API cross reference:

- CR VOS LL_PWR_GetRegulVoltageScaling

LL_PWR_EnableFlashPowerDown

Function name

__STATIC_INLINE void LL_PWR_EnableFlashPowerDown (void)

Function description

Enable the Flash Power Down in Stop Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR FPDS LL_PWR_EnableFlashPowerDown

LL_PWR_DisableFlashPowerDown

Function name

__STATIC_INLINE void LL_PWR_DisableFlashPowerDown (void)

Function description

Disable the Flash Power Down in Stop Mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR FPDS LL_PWR_DisableFlashPowerDown

LL_PWR_IsEnabledFlashPowerDown

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledFlashPowerDown (void)

Function description

Check if the Flash Power Down in Stop Mode is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR FPDS LL_PWR_IsEnabledFlashPowerDown

LL_PWR_EnableBkUpAccess

Function name

__STATIC_INLINE void LL_PWR_EnableBkUpAccess (void)

Function description

Enable access to the backup domain.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DBP LL_PWR_EnableBkUpAccess

LL_PWR_DisableBkUpAccess

Function name

__STATIC_INLINE void LL_PWR_DisableBkUpAccess (void)

Function description

Disable access to the backup domain.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR DBP LL_PWR_DisableBkUpAccess

LL_PWR_IsEnabledBkUpAccess

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledBkUpAccess (void)

Function description

Check if the backup domain is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR DBP LL_PWR_IsEnabledBkUpAccess

LL_PWR_EnableBkUpRegulator

Function name

__STATIC_INLINE void LL_PWR_EnableBkUpRegulator (void)

Function description

Enable the backup Regulator.

Return values

- **None:**

Notes

- The BRE bit of the PWR_CSR register is protected against parasitic write access. The LL_PWR_EnableBkUpAccess() must be called before using this API.

Reference Manual to LL API cross reference:

- CSR BRE LL_PWR_EnableBkUpRegulator

LL_PWR_DisableBkUpRegulator

Function name

__STATIC_INLINE void LL_PWR_DisableBkUpRegulator (void)

Function description

Disable the backup Regulator.

Return values

- **None:**

Notes

- The BRE bit of the PWR_CSR register is protected against parasitic write access. The LL_PWR_EnableBkUpAccess() must be called before using this API.

Reference Manual to LL API cross reference:

- CSR BRE LL_PWR_DisableBkUpRegulator

LL_PWR_IsEnabledBkUpRegulator

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledBkUpRegulator (void)

Function description

Check if the backup Regulator is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR BRE LL_PWR_IsEnabledBkUpRegulator

LL_PWR_SetRegulModeDS

Function name

__STATIC_INLINE void LL_PWR_SetRegulModeDS (uint32_t RegulMode)

Function description

Set voltage Regulator mode during deep sleep mode.

Parameters

- **RegulMode:** This parameter can be one of the following values:
  - LL_PWR_REGU_DSMODE_MAIN
  - LL_PWR_REGU_DSMODE_LOW_POWER

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR LPDS LL_PWR_SetRegulModeDS

LL_PWR_GetRegulModeDS

Function name

__STATIC_INLINE uint32_t LL_PWR_GetRegulModeDS (void)

Function description

Get voltage Regulator mode during deep sleep mode.

Return values

- **Returned:** value can be one of the following values:
  - LL_PWR_REGU_DSMODE_MAIN
  - LL_PWR_REGU_DSMODE_LOW_POWER

Reference Manual to LL API cross reference:

- CR LPDS LL_PWR_GetRegulModeDS

LL_PWR_SetPowerMode

Function name

__STATIC_INLINE void LL_PWR_SetPowerMode (uint32_t PDMode)

Function description

Set Power Down mode when CPU enters deepsleep.

Parameters

- **PDMode:** This parameter can be one of the following values:
  - LL_PWR_MODE_STOP_MAINREGU
  - LL_PWR_MODE_STOP_LPREGU
  - LL_PWR_MODE_STOP_MAINREGU_UNDERDRIVE (*)
  - LL_PWR_MODE_STOP_LPREGU_UNDERDRIVE (*)
  - LL_PWR_MODE_STOP_MAINREGU_DEEPSLEEP (*)
  - LL_PWR_MODE_STOP_LPREGU_DEEPSLEEP (*)

  (*) not available on all devices
  - LL_PWR_MODE_STANDBY

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PDDS LL_PWR_SetPowerMode
-
- CR MRUDS LL_PWR_SetPowerMode
-
- CR LPUDS LL_PWR_SetPowerMode
-
- CR FPDS LL_PWR_SetPowerMode
-
- CR MRLVDS LL_PWR_SetPowerMode
-
- CR LPIVDS LL_PWR_SetPowerMode
-
- CR FPDS LL_PWR_SetPowerMode
-
- CR LPDS LL_PWR_SetPowerMode

LL_PWR_GetPowerMode

Function name

__STATIC_INLINE uint32_t LL_PWR_GetPowerMode (void)

Function description

Get Power Down mode when CPU enters deepsleep.

Return values

- **Returned:** value can be one of the following values:
  - LL_PWR_MODE_STOP_MAINREGU
  - LL_PWR_MODE_STOP_LPREGU
  - LL_PWR_MODE_STOP_MAINREGU_UNDERDRIVE (*)
  - LL_PWR_MODE_STOP_LPREGU_UNDERDRIVE (*)
  - LL_PWR_MODE_STOP_MAINREGU_DEEPSLEEP (*)
  - LL_PWR_MODE_STOP_LPREGU_DEEPSLEEP (*)

  (*) not available on all devices
  - LL_PWR_MODE_STANDBY

Reference Manual to LL API cross reference:

- CR PDDS LL_PWR_GetPowerMode
-
- CR MRUDS LL_PWR_GetPowerMode
-
- CR LPUDS LL_PWR_GetPowerMode
-
- CR FPDS LL_PWR_GetPowerMode
-
- CR MRLVDS LL_PWR_GetPowerMode
-
- CR LPLVDS LL_PWR_GetPowerMode
-
- CR FPDS LL_PWR_GetPowerMode
-
- CR LPDS LL_PWR_GetPowerMode

LL_PWR_SetPVDLevel

Function name

__STATIC_INLINE void LL_PWR_SetPVDLevel (uint32_t PVDLevel)

Function description

Configure the voltage threshold detected by the Power Voltage Detector.

Parameters

- **PVDLevel:** This parameter can be one of the following values:
  - LL_PWR_PVDLEVEL_0
  - LL_PWR_PVDLEVEL_1
  - LL_PWR_PVDLEVEL_2
  - LL_PWR_PVDLEVEL_3
  - LL_PWR_PVDLEVEL_4
  - LL_PWR_PVDLEVEL_5
  - LL_PWR_PVDLEVEL_6
  - LL_PWR_PVDLEVEL_7

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PLS LL_PWR_SetPVDLevel

LL_PWR_GetPVDLevel

Function name

__STATIC_INLINE uint32_t LL_PWR_GetPVDLevel (void)

Function description

Get the voltage threshold detection.

Return values

- **Returned:** value can be one of the following values:
  - LL_PWR_PVDLEVEL_0
  - LL_PWR_PVDLEVEL_1
  - LL_PWR_PVDLEVEL_2
  - LL_PWR_PVDLEVEL_3
  - LL_PWR_PVDLEVEL_4
  - LL_PWR_PVDLEVEL_5
  - LL_PWR_PVDLEVEL_6
  - LL_PWR_PVDLEVEL_7

Reference Manual to LL API cross reference:

- CR PLS LL_PWR_GetPVDLevel

LL_PWR_EnablePVD

Function name

__STATIC_INLINE void LL_PWR_EnablePVD (void)

Function description

Enable Power Voltage Detector.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PVDE LL_PWR_EnablePVD

LL_PWR_DisablePVD

Function name

__STATIC_INLINE void LL_PWR_DisablePVD (void)

Function description

Disable Power Voltage Detector.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PVDE LL_PWR_DisablePVD

LL_PWR_IsEnabledPVD

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledPVD (void)

Function description

Check if Power Voltage Detector is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR PVDE LL_PWR_IsEnabledPVD

LL_PWR_EnableWakeUpPin

Function name

__STATIC_INLINE void LL_PWR_EnableWakeUpPin (uint32_t WakeUpPin)

Function description

Enable the WakeUp PINx functionality.

Parameters

- **WakeUpPin:** This parameter can be one of the following values:
  - LL_PWR_WAKEUP_PIN1
  - LL_PWR_WAKEUP_PIN2 (*)
  - LL_PWR_WAKEUP_PIN3 (*)

  (*) not available on all devices

Return values

- **None:**

Reference Manual to LL API cross reference:

- CSR EWUP LL_PWR_EnableWakeUpPin
-
- CSR EWUP1 LL_PWR_EnableWakeUpPin
-
- CSR EWUP2 LL_PWR_EnableWakeUpPin
-
- CSR EWUP3 LL_PWR_EnableWakeUpPin

LL_PWR_DisableWakeUpPin

Function name

__STATIC_INLINE void LL_PWR_DisableWakeUpPin (uint32_t WakeUpPin)

Function description

Disable the WakeUp PINx functionality.

Parameters

- **WakeUpPin:** This parameter can be one of the following values:
  - LL_PWR_WAKEUP_PIN1
  - LL_PWR_WAKEUP_PIN2 (*)
  - LL_PWR_WAKEUP_PIN3 (*)

  (*) not available on all devices

Return values

- **None:**

Reference Manual to LL API cross reference:

- CSR EWUP LL_PWR_DisableWakeUpPin
-
- CSR EWUP1 LL_PWR_DisableWakeUpPin
-
- CSR EWUP2 LL_PWR_DisableWakeUpPin
-
- CSR EWUP3 LL_PWR_DisableWakeUpPin

LL_PWR_IsEnabledWakeUpPin

Function name

__STATIC_INLINE uint32_t LL_PWR_IsEnabledWakeUpPin (uint32_t WakeUpPin)

Function description

Check if the WakeUp PINx functionality is enabled.

Parameters

- **WakeUpPin:** This parameter can be one of the following values:
  - LL_PWR_WAKEUP_PIN1
  - LL_PWR_WAKEUP_PIN2 (*)
  - LL_PWR_WAKEUP_PIN3 (*)

  (*) not available on all devices

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR EWUP LL_PWR_IsEnabledWakeUpPin
-
- CSR EWUP1 LL_PWR_IsEnabledWakeUpPin
-
- CSR EWUP2 LL_PWR_IsEnabledWakeUpPin
-
- CSR EWUP3 LL_PWR_IsEnabledWakeUpPin

LL_PWR_IsActiveFlag_WU

Function name

__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_WU (void)

Function description

Get Wake-up Flag.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR WUF LL_PWR_IsActiveFlag_WU

LL_PWR_IsActiveFlag_SB

Function name

__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_SB (void)

Function description

Get Standby Flag.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR SBF LL_PWR_IsActiveFlag_SB

LL_PWR_IsActiveFlag_BRR

Function name

__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_BRR (void)

Function description

Get Backup Regulator ready Flag.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR BRR LL_PWR_IsActiveFlag_BRR

LL_PWR_IsActiveFlag_PVDO

Function name

__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_PVDO (void)

Function description

Indicate whether VDD voltage is below the selected PVD threshold.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR PVDO LL_PWR_IsActiveFlag_PVDO

LL_PWR_IsActiveFlag_VOS

Function name

__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_VOS (void)

Function description

Indicate whether the Regulator is ready in the selected voltage range or if its output voltage is still changing to the required voltage level.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR VOS LL_PWR_IsActiveFlag_VOS

LL_PWR_IsActiveFlag_OD

Function name

__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_OD (void)

Function description

Indicate whether the Over-Drive mode is ready or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR ODRDY LL_PWR_IsActiveFlag_OD

LL_PWR_IsActiveFlag_ODSW

Function name

__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_ODSW (void)

Function description

Indicate whether the Over-Drive mode switching is ready or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR ODSWRDY LL_PWR_IsActiveFlag_ODSW

LL_PWR_IsActiveFlag_UD

Function name

__STATIC_INLINE uint32_t LL_PWR_IsActiveFlag_UD (void)

Function description

Indicate whether the Under-Drive mode is ready or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR UDRDY LL_PWR_IsActiveFlag_UD

LL_PWR_ClearFlag_SB

Function name

__STATIC_INLINE void LL_PWR_ClearFlag_SB (void)

Function description

Clear Standby Flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CSBF LL_PWR_ClearFlag_SB

LL_PWR_ClearFlag_WU

Function name

__STATIC_INLINE void LL_PWR_ClearFlag_WU (void)

Function description

Clear Wake-up Flags.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CWUF LL_PWR_ClearFlag_WU

LL_PWR_ClearFlag_UD

Function name

__STATIC_INLINE void LL_PWR_ClearFlag_UD (void)

Function description

Clear Under-Drive ready Flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CSR UDRDY LL_PWR_ClearFlag_UD

LL_PWR_DeInit

Function name

ErrorStatus LL_PWR_DeInit (void)

Function description

De-initialize the PWR registers to their default reset values.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: PWR registers are de-initialized
  - ERROR: not applicable

86.2 PWR Firmware driver defines

The following section lists the various define and macros of the module.

86.2.1 PWR

PWR

Clear Flags Defines

LL_PWR_CR_CSBF

Clear standby flag

LL_PWR_CR_CWUF

Clear wakeup flag

Get Flags Defines

LL_PWR_CSR_WUF

Wakeup flag

LL_PWR_CSR_SBF

Standby flag

LL_PWR_CSR_PVDO

Power voltage detector output flag

LL_PWR_CSR_VOS

Voltage scaling select flag

LL_PWR_CSR_EWUP1

Enable WKUP pin

Mode Power

LL_PWR_MODE_STOP_MAINREGU

Enter Stop mode when the CPU enters deepsleep

LL_PWR_MODE_STOP_LPREGU

Enter Stop mode (with low power Regulator ON) when the CPU enters deepsleep

LL_PWR_MODE_STOP_MAINREGU_UNDERDRIVE

Enter Stop mode (with main Regulator in under-drive mode) when the CPU enters deepsleep

LL_PWR_MODE_STOP_LPREGU_UNDERDRIVE

Enter Stop mode (with low power Regulator in under-drive mode) when the CPU enters deepsleep

LL_PWR_MODE_STOP_MAINREGU_DEEPSLEEP

Enter Stop mode (with main Regulator in Deep Sleep mode) when the CPU enters deepsleep

LL_PWR_MODE_STOP_LPREGU_DEEPSLEEP

Enter Stop mode (with low power Regulator in Deep Sleep mode) when the CPU enters deepsleep

LL_PWR_MODE_STANDBY

Enter Standby mode when the CPU enters deepsleep

Power Voltage Detector Level

LL_PWR_PVDLEVEL_0

Voltage threshold detected by PVD 2.2 V

LL_PWR_PVDLEVEL_1

Voltage threshold detected by PVD 2.3 V

LL_PWR_PVDLEVEL_2

Voltage threshold detected by PVD 2.4 V

LL_PWR_PVDLEVEL_3

Voltage threshold detected by PVD 2.5 V

LL_PWR_PVDLEVEL_4

Voltage threshold detected by PVD 2.6 V

LL_PWR_PVDLEVEL_5

Voltage threshold detected by PVD 2.7 V

LL_PWR_PVDLEVEL_6

Voltage threshold detected by PVD 2.8 V

LL_PWR_PVDLEVEL_7

Voltage threshold detected by PVD 2.9 V

Regulator Mode In Deep Sleep Mode

LL_PWR_REGU_DSMODE_MAIN

Voltage Regulator in main mode during deepsleep mode

LL_PWR_REGU_DSMODE_LOW_POWER

Voltage Regulator in low-power mode during deepsleep mode

Regulator Voltage

LL_PWR_REGU_VOLTAGE_SCALE3

LL_PWR_REGU_VOLTAGE_SCALE2

LL_PWR_REGU_VOLTAGE_SCALE1

Wakeup Pins

LL_PWR_WAKEUP_PIN1

WKUP pin : PA0

Common write and read registers Macros

LL_PWR_WriteReg

Description:

- Write a value in PWR register.

Parameters:

- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_PWR_ReadReg

Description:

- Read a value in PWR register.

Parameters:

- __REG__: Register to be read

Return value:

- Register: value

87 LL RCC Generic Driver

87.1 RCC Firmware driver registers structures

87.1.1 LL_RCC_ClocksTypeDef

LL_RCC_ClocksTypeDef is defined in the stm32f4xx_ll_rcc.h

Data Fields

- ***uint32_t SYSCLK_Frequency***
- ***uint32_t HCLK_Frequency***
- ***uint32_t PCLK1_Frequency***
- ***uint32_t PCLK2_Frequency***

Field Documentation

- ***uint32_t LL_RCC_ClocksTypeDef::SYSCLK_Frequency***
  SYSCLK clock frequency
- ***uint32_t LL_RCC_ClocksTypeDef::HCLK_Frequency***
  HCLK clock frequency
- ***uint32_t LL_RCC_ClocksTypeDef::PCLK1_Frequency***
  PCLK1 clock frequency
- ***uint32_t LL_RCC_ClocksTypeDef::PCLK2_Frequency***
  PCLK2 clock frequency

87.2 RCC Firmware driver API description

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

87.2.1 Detailed description of functions

LL_RCC_HSE_EnableCSS

Function name

__STATIC_INLINE void LL_RCC_HSE_EnableCSS (void)

Function description

Enable the Clock Security System.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR CSSON LL_RCC_HSE_EnableCSS

LL_RCC_HSE_EnableBypass

Function name

__STATIC_INLINE void LL_RCC_HSE_EnableBypass (void)

Function description

Enable HSE external oscillator (HSE Bypass)

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR HSEBYP LL_RCC_HSE_EnableBypass

LL_RCC_HSE_DisableBypass

Function name

__STATIC_INLINE void LL_RCC_HSE_DisableBypass (void)

Function description

Disable HSE external oscillator (HSE Bypass)

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR HSEBYP LL_RCC_HSE_DisableBypass

LL_RCC_HSE_Enable

Function name

__STATIC_INLINE void LL_RCC_HSE_Enable (void)

Function description

Enable HSE crystal oscillator (HSE ON)

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR HSEON LL_RCC_HSE_Enable

LL_RCC_HSE_Disable

Function name

__STATIC_INLINE void LL_RCC_HSE_Disable (void)

Function description

Disable HSE crystal oscillator (HSE ON)

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR HSEON LL_RCC_HSE_Disable

LL_RCC_HSE_IsReady

Function name

__STATIC_INLINE uint32_t LL_RCC_HSE_IsReady (void)

Function description

Check if HSE oscillator Ready.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR HSERDY LL_RCC_HSE_IsReady

LL_RCC_HSI_Enable

Function name

__STATIC_INLINE void LL_RCC_HSI_Enable (void)

Function description

Enable HSI oscillator.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR HSION LL_RCC_HSI_Enable

LL_RCC_HSI_Disable

Function name

__STATIC_INLINE void LL_RCC_HSI_Disable (void)

Function description

Disable HSI oscillator.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR HSION LL_RCC_HSI_Disable

LL_RCC_HSI_IsReady

Function name

__STATIC_INLINE uint32_t LL_RCC_HSI_IsReady (void)

Function description

Check if HSI clock is ready.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR HSIRDY LL_RCC_HSI_IsReady

LL_RCC_HSI_GetCalibration

Function name

__STATIC_INLINE uint32_t LL_RCC_HSI_GetCalibration (void)

Function description

Get HSI Calibration value.

Return values

- **Between:** Min_Data = 0x00 and Max_Data = 0xFF

Notes

- When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value

Reference Manual to LL API cross reference:

- CR HSICAL LL_RCC_HSI_GetCalibration

LL_RCC_HSI_SetCalibTrimming

Function name

__STATIC_INLINE void LL_RCC_HSI_SetCalibTrimming (uint32_t Value)

Function description

Set HSI Calibration trimming.

Parameters

- **Value:** Between Min_Data = 0 and Max_Data = 31

Return values

- **None:**

Notes

- user-programmable trimming value that is added to the HSICAL
- Default value is 16, which, when added to the HSICAL value, should trim the HSI to 16 MHz +/- 1 %

Reference Manual to LL API cross reference:

- CR HSITRIM LL_RCC_HSI_SetCalibTrimming

LL_RCC_HSI_GetCalibTrimming

Function name

__STATIC_INLINE uint32_t LL_RCC_HSI_GetCalibTrimming (void)

Function description

Get HSI Calibration trimming.

Return values

- **Between:** Min_Data = 0 and Max_Data = 31

Reference Manual to LL API cross reference:

- CR HSITRIM LL_RCC_HSI_GetCalibTrimming

LL_RCC_LSE_Enable

Function name

__STATIC_INLINE void LL_RCC_LSE_Enable (void)

Function description

Enable Low Speed External (LSE) crystal.

Return values

- **None:**

Reference Manual to LL API cross reference:

- BDCR LSEON LL_RCC_LSE_Enable

LL_RCC_LSE_Disable

Function name

__STATIC_INLINE void LL_RCC_LSE_Disable (void)

Function description

Disable Low Speed External (LSE) crystal.

Return values

- **None:**

Reference Manual to LL API cross reference:

- BDCR LSEON LL_RCC_LSE_Disable

LL_RCC_LSE_EnableBypass

Function name

__STATIC_INLINE void LL_RCC_LSE_EnableBypass (void)

Function description

Enable external clock source (LSE bypass).

Return values

- **None:**

Reference Manual to LL API cross reference:

- BDCR LSEBYP LL_RCC_LSE_EnableBypass

LL_RCC_LSE_DisableBypass

Function name

__STATIC_INLINE void LL_RCC_LSE_DisableBypass (void)

Function description

Disable external clock source (LSE bypass).

Return values

- **None:**

Reference Manual to LL API cross reference:

- BDCR LSEBYP LL_RCC_LSE_DisableBypass

LL_RCC_LSE_IsReady

Function name

__STATIC_INLINE uint32_t LL_RCC_LSE_IsReady (void)

Function description

Check if LSE oscillator Ready.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- BDCR LSERDY LL_RCC_LSE_IsReady

LL_RCC_LSE_EnableHighDriveMode

Function name

__STATIC_INLINE void LL_RCC_LSE_EnableHighDriveMode (void)

Function description

Enable LSE high drive mode.

Return values

- **None:**

Notes

- LSE high drive mode can be enabled only when the LSE clock is disabled

Reference Manual to LL API cross reference:

- BDCR LSEMOD LL_RCC_LSE_EnableHighDriveMode

LL_RCC_LSE_DisableHighDriveMode

Function name

__STATIC_INLINE void LL_RCC_LSE_DisableHighDriveMode (void)

Function description

Disable LSE high drive mode.

Return values

- **None:**

Notes

- LSE high drive mode can be disabled only when the LSE clock is disabled

Reference Manual to LL API cross reference:

- BDCR LSEMOD LL_RCC_LSE_DisableHighDriveMode

LL_RCC_LSI_Enable

Function name

__STATIC_INLINE void LL_RCC_LSI_Enable (void)

Function description

Enable LSI Oscillator.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CSR LSION LL_RCC_LSI_Enable

LL_RCC_LSI_Disable

Function name

__STATIC_INLINE void LL_RCC_LSI_Disable (void)

Function description

Disable LSI Oscillator.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CSR LSION LL_RCC_LSI_Disable

LL_RCC_LSI_IsReady

Function name

__STATIC_INLINE uint32_t LL_RCC_LSI_IsReady (void)

Function description

Check if LSI is Ready.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR LSIRDY LL_RCC_LSI_IsReady

LL_RCC_SetSysClkSource

Function name

__STATIC_INLINE void LL_RCC_SetSysClkSource (uint32_t Source)

Function description

Configure the system clock source.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_SYS_CLKSOURCE_HSI
  - LL_RCC_SYS_CLKSOURCE_HSE
  - LL_RCC_SYS_CLKSOURCE_PLL
  - LL_RCC_SYS_CLKSOURCE_PLLR (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR SW LL_RCC_SetSysClkSource

LL_RCC_GetSysClkSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetSysClkSource (void)

Function description

Get the system clock source.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_SYS_CLKSOURCE_STATUS_HSI
  - LL_RCC_SYS_CLKSOURCE_STATUS_HSE
  - LL_RCC_SYS_CLKSOURCE_STATUS_PLL
  - LL_RCC_SYS_CLKSOURCE_STATUS_PLLR (*)

  (*) value not defined in all devices.

Reference Manual to LL API cross reference:

- CFGR SWS LL_RCC_GetSysClkSource

LL_RCC_SetAHBPrescaler

Function name

__STATIC_INLINE void LL_RCC_SetAHBPrescaler (uint32_t Prescaler)

Function description

Set AHB prescaler.

Parameters

- **Prescaler:** This parameter can be one of the following values:
  - LL_RCC_SYSCLK_DIV_1
  - LL_RCC_SYSCLK_DIV_2
  - LL_RCC_SYSCLK_DIV_4
  - LL_RCC_SYSCLK_DIV_8
  - LL_RCC_SYSCLK_DIV_16
  - LL_RCC_SYSCLK_DIV_64
  - LL_RCC_SYSCLK_DIV_128
  - LL_RCC_SYSCLK_DIV_256
  - LL_RCC_SYSCLK_DIV_512

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR HPRE LL_RCC_SetAHBPrescaler

LL_RCC_SetAPB1Prescaler

Function name

__STATIC_INLINE void LL_RCC_SetAPB1Prescaler (uint32_t Prescaler)

Function description

Set APB1 prescaler.

Parameters

- **Prescaler:** This parameter can be one of the following values:
  - LL_RCC_APB1_DIV_1
  - LL_RCC_APB1_DIV_2
  - LL_RCC_APB1_DIV_4
  - LL_RCC_APB1_DIV_8
  - LL_RCC_APB1_DIV_16

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR PPRE1 LL_RCC_SetAPB1Prescaler

LL_RCC_SetAPB2Prescaler

Function name

__STATIC_INLINE void LL_RCC_SetAPB2Prescaler (uint32_t Prescaler)

Function description

Set APB2 prescaler.

Parameters

- **Prescaler:** This parameter can be one of the following values:
  - LL_RCC_APB2_DIV_1
  - LL_RCC_APB2_DIV_2
  - LL_RCC_APB2_DIV_4
  - LL_RCC_APB2_DIV_8
  - LL_RCC_APB2_DIV_16

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR PPRE2 LL_RCC_SetAPB2Prescaler

LL_RCC_GetAHBPrescaler

Function name

__STATIC_INLINE uint32_t LL_RCC_GetAHBPrescaler (void)

Function description

Get AHB prescaler.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_SYSCLK_DIV_1
  - LL_RCC_SYSCLK_DIV_2
  - LL_RCC_SYSCLK_DIV_4
  - LL_RCC_SYSCLK_DIV_8
  - LL_RCC_SYSCLK_DIV_16
  - LL_RCC_SYSCLK_DIV_64
  - LL_RCC_SYSCLK_DIV_128
  - LL_RCC_SYSCLK_DIV_256
  - LL_RCC_SYSCLK_DIV_512

Reference Manual to LL API cross reference:

- CFGR HPRE LL_RCC_GetAHBPrescaler

LL_RCC_GetAPB1Prescaler

Function name

__STATIC_INLINE uint32_t LL_RCC_GetAPB1Prescaler (void)

Function description

Get APB1 prescaler.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_APB1_DIV_1
  - LL_RCC_APB1_DIV_2
  - LL_RCC_APB1_DIV_4
  - LL_RCC_APB1_DIV_8
  - LL_RCC_APB1_DIV_16

Reference Manual to LL API cross reference:

- CFGR PPRE1 LL_RCC_GetAPB1Prescaler

LL_RCC_GetAPB2Prescaler

Function name

__STATIC_INLINE uint32_t LL_RCC_GetAPB2Prescaler (void)

Function description

Get APB2 prescaler.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_APB2_DIV_1
  - LL_RCC_APB2_DIV_2
  - LL_RCC_APB2_DIV_4
  - LL_RCC_APB2_DIV_8
  - LL_RCC_APB2_DIV_16

Reference Manual to LL API cross reference:

- CFGR PPRE2 LL_RCC_GetAPB2Prescaler

LL_RCC_ConfigMCO

Function name

__STATIC_INLINE void LL_RCC_ConfigMCO (uint32_t MCOxSource, uint32_t MCOxPrescaler)

Function description

Configure MCOx.

Parameters

- **MCOxSource:** This parameter can be one of the following values:
  - LL_RCC_MCO1SOURCE_HSI
  - LL_RCC_MCO1SOURCE_LSE
  - LL_RCC_MCO1SOURCE_HSE
  - LL_RCC_MCO1SOURCE_PLLCLK
  - LL_RCC_MCO2SOURCE_SYSCLK
  - LL_RCC_MCO2SOURCE_PLLI2S
  - LL_RCC_MCO2SOURCE_HSE
  - LL_RCC_MCO2SOURCE_PLLCLK
- **MCOxPrescaler:** This parameter can be one of the following values:
  - LL_RCC_MCO1_DIV_1
  - LL_RCC_MCO1_DIV_2
  - LL_RCC_MCO1_DIV_3
  - LL_RCC_MCO1_DIV_4
  - LL_RCC_MCO1_DIV_5
  - LL_RCC_MCO2_DIV_1
  - LL_RCC_MCO2_DIV_2
  - LL_RCC_MCO2_DIV_3
  - LL_RCC_MCO2_DIV_4
  - LL_RCC_MCO2_DIV_5

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR MCO1 LL_RCC_ConfigMCO
- CFGR MCO1PRE LL_RCC_ConfigMCO
- CFGR MCO2 LL_RCC_ConfigMCO
- CFGR MCO2PRE LL_RCC_ConfigMCO

LL_RCC_SetSAIClockSource

Function name

__STATIC_INLINE void LL_RCC_SetSAIClockSource (uint32_t SAIxSource)

Function description

Configure SAIx clock source.

Parameters

- **SAIxSource:** This parameter can be one of the following values:
  - LL_RCC_SAI1_CLKSOURCE_PLLSAI (*)
  - LL_RCC_SAI1_CLKSOURCE_PLLI2S (*)
  - LL_RCC_SAI1_CLKSOURCE_PLL (*)
  - LL_RCC_SAI1_CLKSOURCE_PIN (*)
  - LL_RCC_SAI2_CLKSOURCE_PLLSAI (*)
  - LL_RCC_SAI2_CLKSOURCE_PLLI2S (*)
  - LL_RCC_SAI2_CLKSOURCE_PLL (*)
  - LL_RCC_SAI2_CLKSOURCE_PLLSRC (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PLLSAI (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PLLI2S (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PIN (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PLL (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PLLSRC (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PLLSAI (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PLLI2S (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PIN (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PLL (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PLLSRC (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DCKCFGR SAI1SRC LL_RCC_SetSAIClockSource
- DCKCFGR SAI2SRC LL_RCC_SetSAIClockSource
- DCKCFGR SAI1ASRC LL_RCC_SetSAIClockSource
- DCKCFGR SAI1BSRC LL_RCC_SetSAIClockSource

LL_RCC_SetSDIOClockSource

Function name

__STATIC_INLINE void LL_RCC_SetSDIOClockSource (uint32_t SDIOxSource)

Function description

Configure SDIO clock source.

Parameters

- **SDIOxSource:** This parameter can be one of the following values:
  - LL_RCC_SDIO_CLKSOURCE_PLL48CLK
  - LL_RCC_SDIO_CLKSOURCE_SYSCLK

Return values

- **None:**

Reference Manual to LL API cross reference:

- DCKCFGR SDIOSEL LL_RCC_SetSDIOClockSource
- DCKCFGR2 SDIOSEL LL_RCC_SetSDIOClockSource

LL_RCC_SetCK48MClockSource

Function name

__STATIC_INLINE void LL_RCC_SetCK48MClockSource (uint32_t CK48MxSource)

Function description

Configure 48Mhz domain clock source.

Parameters

- **CK48MxSource:** This parameter can be one of the following values:
  - LL_RCC_CK48M_CLKSOURCE_PLL
  - LL_RCC_CK48M_CLKSOURCE_PLLSAI (*)
  - LL_RCC_CK48M_CLKSOURCE_PLLI2S (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DCKCFGR CK48MSEL LL_RCC_SetCK48MClockSource
- DCKCFGR2 CK48MSEL LL_RCC_SetCK48MClockSource

LL_RCC_SetRNGClockSource

Function name

__STATIC_INLINE void LL_RCC_SetRNGClockSource (uint32_t RNGxSource)

Function description

Configure RNG clock source.

Parameters

- **RNGxSource:** This parameter can be one of the following values:
  - LL_RCC_RNG_CLKSOURCE_PLL
  - LL_RCC_RNG_CLKSOURCE_PLLSAI (*)
  - LL_RCC_RNG_CLKSOURCE_PLLI2S (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DCKCFGR CK48MSEL LL_RCC_SetRNGClockSource
- DCKCFGR2 CK48MSEL LL_RCC_SetRNGClockSource

LL_RCC_SetUSBClockSource

Function name

__STATIC_INLINE void LL_RCC_SetUSBClockSource (uint32_t USBxSource)

Function description

Configure USB clock source.

Parameters

- **USBxSource:** This parameter can be one of the following values:
  - LL_RCC_USB_CLKSOURCE_PLL
  - LL_RCC_USB_CLKSOURCE_PLLSAI (*)
  - LL_RCC_USB_CLKSOURCE_PLLI2S (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DCKCFGR CK48MSEL LL_RCC_SetUSBClockSource
- DCKCFGR2 CK48MSEL LL_RCC_SetUSBClockSource

LL_RCC_SetI2SClockSource

Function name

__STATIC_INLINE void LL_RCC_SetI2SClockSource (uint32_t Source)

Function description

Configure I2S clock source.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_I2S1_CLKSOURCE_PLLI2S (*)
  - LL_RCC_I2S1_CLKSOURCE_PIN
  - LL_RCC_I2S1_CLKSOURCE_PLL (*)
  - LL_RCC_I2S1_CLKSOURCE_PLLSRC (*)
  - LL_RCC_I2S2_CLKSOURCE_PLLI2S (*)
  - LL_RCC_I2S2_CLKSOURCE_PIN (*)
  - LL_RCC_I2S2_CLKSOURCE_PLL (*)
  - LL_RCC_I2S2_CLKSOURCE_PLLSRC (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR I2SSRC LL_RCC_SetI2SClockSource
- DCKCFGR I2SSRC LL_RCC_SetI2SClockSource
- DCKCFGR I2S1SRC LL_RCC_SetI2SClockSource
- DCKCFGR I2S2SRC LL_RCC_SetI2SClockSource

LL_RCC_SetDSIClockSource

Function name

__STATIC_INLINE void LL_RCC_SetDSIClockSource (uint32_t Source)

Function description

Configure DSI clock source.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_DSI_CLKSOURCE_PHY
  - LL_RCC_DSI_CLKSOURCE_PLL

Return values

- **None:**

Reference Manual to LL API cross reference:

- DCKCFGR DSISEL LL_RCC_SetDSIClockSource

LL_RCC_GetSAIClockSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetSAIClockSource (uint32_t SAIx)

Function description

Get SAIx clock source.

Parameters

- **SAIx:** This parameter can be one of the following values:
  - LL_RCC_SAI1_CLKSOURCE (*)
  - LL_RCC_SAI2_CLKSOURCE (*)
  - LL_RCC_SAI1_A_CLKSOURCE (*)
  - LL_RCC_SAI1_B_CLKSOURCE (*)

  (*) value not defined in all devices.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_SAI1_CLKSOURCE_PLLSAI (*)
  - LL_RCC_SAI1_CLKSOURCE_PLLI2S (*)
  - LL_RCC_SAI1_CLKSOURCE_PLL (*)
  - LL_RCC_SAI1_CLKSOURCE_PIN (*)
  - LL_RCC_SAI2_CLKSOURCE_PLLSAI (*)
  - LL_RCC_SAI2_CLKSOURCE_PLLI2S (*)
  - LL_RCC_SAI2_CLKSOURCE_PLL (*)
  - LL_RCC_SAI2_CLKSOURCE_PLLSRC (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PLLSAI (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PLLI2S (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PIN (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PLL (*)
  - LL_RCC_SAI1_A_CLKSOURCE_PLLSRC (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PLLSAI (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PLLI2S (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PIN (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PLL (*)
  - LL_RCC_SAI1_B_CLKSOURCE_PLLSRC (*)

  (*) value not defined in all devices.

Reference Manual to LL API cross reference:

- DCKCFGR SAI1SEL LL_RCC_GetSAIClockSource
- DCKCFGR SAI2SEL LL_RCC_GetSAIClockSource
- DCKCFGR SAI1ASRC LL_RCC_GetSAIClockSource
- DCKCFGR SAI1BSRC LL_RCC_GetSAIClockSource

LL_RCC_GetSDIOClockSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetSDIOClockSource (uint32_t SDIOx)

Function description

Get SDIOx clock source.

Parameters

- **SDIOx:** This parameter can be one of the following values:
  - LL_RCC_SDIO_CLKSOURCE

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_SDIO_CLKSOURCE_PLL48CLK
  - LL_RCC_SDIO_CLKSOURCE_SYSCLK

Reference Manual to LL API cross reference:

- DCKCFGR SDIOSEL LL_RCC_GetSDIOClockSource
- DCKCFGR2 SDIOSEL LL_RCC_GetSDIOClockSource

LL_RCC_GetCK48MClockSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetCK48MClockSource (uint32_t CK48Mx)

Function description

Get 48Mhz domain clock source.

Parameters

- **CK48Mx:** This parameter can be one of the following values:
  - LL_RCC_CK48M_CLKSOURCE

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_CK48M_CLKSOURCE_PLL
  - LL_RCC_CK48M_CLKSOURCE_PLLSAI (*)
  - LL_RCC_CK48M_CLKSOURCE_PLLI2S (*)

  (*) value not defined in all devices.

Reference Manual to LL API cross reference:

- DCKCFGR CK48MSEL LL_RCC_GetCK48MClockSource
- DCKCFGR2 CK48MSEL LL_RCC_GetCK48MClockSource

LL_RCC_GetRNGClockSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetRNGClockSource (uint32_t RNGx)

Function description

Get RNGx clock source.

Parameters

- **RNGx:** This parameter can be one of the following values:
  - LL_RCC_RNG_CLKSOURCE

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_RNG_CLKSOURCE_PLL
  - LL_RCC_RNG_CLKSOURCE_PLLSAI (*)
  - LL_RCC_RNG_CLKSOURCE_PLLI2S (*)

  (*) value not defined in all devices.

Reference Manual to LL API cross reference:

- DCKCFGR CK48MSEL LL_RCC_GetRNGClockSource
- DCKCFGR2 CK48MSEL LL_RCC_GetRNGClockSource

LL_RCC_GetUSBClockSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetUSBClockSource (uint32_t USBx)

Function description

Get USBx clock source.

Parameters

- **USBx:** This parameter can be one of the following values:
  - LL_RCC_USB_CLKSOURCE

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_USB_CLKSOURCE_PLL
  - LL_RCC_USB_CLKSOURCE_PLLSAI (*)
  - LL_RCC_USB_CLKSOURCE_PLLI2S (*)

  (*) value not defined in all devices.

Reference Manual to LL API cross reference:

- DCKCFGR CK48MSEL LL_RCC_GetUSBClockSource
- DCKCFGR2 CK48MSEL LL_RCC_GetUSBClockSource

LL_RCC_GetI2SClockSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetI2SClockSource (uint32_t I2Sx)

Function description

Get I2S Clock Source.

Parameters

- **I2Sx:** This parameter can be one of the following values:
  - LL_RCC_I2S1_CLKSOURCE
  - LL_RCC_I2S2_CLKSOURCE (*)

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_I2S1_CLKSOURCE_PLLI2S (*)
  - LL_RCC_I2S1_CLKSOURCE_PIN
  - LL_RCC_I2S1_CLKSOURCE_PLL (*)
  - LL_RCC_I2S1_CLKSOURCE_PLLSRC (*)
  - LL_RCC_I2S2_CLKSOURCE_PLLI2S (*)
  - LL_RCC_I2S2_CLKSOURCE_PIN (*)
  - LL_RCC_I2S2_CLKSOURCE_PLL (*)
  - LL_RCC_I2S2_CLKSOURCE_PLLSRC (*)

  (*) value not defined in all devices.

Reference Manual to LL API cross reference:

- CFGR I2SSRC LL_RCC_GetI2SClockSource
- DCKCFGR I2SSRC LL_RCC_GetI2SClockSource
- DCKCFGR I2S1SRC LL_RCC_GetI2SClockSource
- DCKCFGR I2S2SRC LL_RCC_GetI2SClockSource

LL_RCC_GetDSIClockSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetDSIClockSource (uint32_t DSIx)

Function description

Get DSI Clock Source.

Parameters

- **DSIx:** This parameter can be one of the following values:
  - LL_RCC_DSI_CLKSOURCE

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_DSI_CLKSOURCE_PHY
  - LL_RCC_DSI_CLKSOURCE_PLL

Reference Manual to LL API cross reference:

- DCKCFGR DSISEL LL_RCC_GetDSIClockSource

LL_RCC_SetRTCClockSource

Function name

__STATIC_INLINE void LL_RCC_SetRTCClockSource (uint32_t Source)

Function description

Set RTC Clock Source.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_RTC_CLKSOURCE_NONE
  - LL_RCC_RTC_CLKSOURCE_LSE
  - LL_RCC_RTC_CLKSOURCE_LSI
  - LL_RCC_RTC_CLKSOURCE_HSE

Return values

- **None:**

Notes

- Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The BDRST bit can be used to reset them.

Reference Manual to LL API cross reference:

- BDCR RTCSEL LL_RCC_SetRTCClockSource

LL_RCC_GetRTCClockSource

Function name

__STATIC_INLINE uint32_t LL_RCC_GetRTCClockSource (void)

Function description

Get RTC Clock Source.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_RTC_CLKSOURCE_NONE
  - LL_RCC_RTC_CLKSOURCE_LSE
  - LL_RCC_RTC_CLKSOURCE_LSI
  - LL_RCC_RTC_CLKSOURCE_HSE

Reference Manual to LL API cross reference:

- BDCR RTCSEL LL_RCC_GetRTCClockSource

LL_RCC_EnableRTC

Function name

__STATIC_INLINE void LL_RCC_EnableRTC (void)

Function description

Enable RTC.

Return values

- **None:**

Reference Manual to LL API cross reference:

- BDCR RTCEN LL_RCC_EnableRTC

LL_RCC_DisableRTC

Function name

__STATIC_INLINE void LL_RCC_DisableRTC (void)

Function description

Disable RTC.

Return values

- **None:**

Reference Manual to LL API cross reference:

- BDCR RTCEN LL_RCC_DisableRTC

LL_RCC_IsEnabledRTC

Function name

__STATIC_INLINE uint32_t LL_RCC_IsEnabledRTC (void)

Function description

Check if RTC has been enabled or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- BDCR RTCEN LL_RCC_IsEnabledRTC

LL_RCC_ForceBackupDomainReset

Function name

__STATIC_INLINE void LL_RCC_ForceBackupDomainReset (void)

Function description

Force the Backup domain reset.

Return values

- **None:**

Reference Manual to LL API cross reference:

- BDCR BDRST LL_RCC_ForceBackupDomainReset

LL_RCC_ReleaseBackupDomainReset

Function name

__STATIC_INLINE void LL_RCC_ReleaseBackupDomainReset (void)

Function description

Release the Backup domain reset.

Return values

- **None:**

Reference Manual to LL API cross reference:

- BDCR BDRST LL_RCC_ReleaseBackupDomainReset

LL_RCC_SetRTC_HSEPrescaler

Function name

__STATIC_INLINE void LL_RCC_SetRTC_HSEPrescaler (uint32_t Prescaler)

Function description

Set HSE Prescalers for RTC Clock.

Parameters

- **Prescaler:** This parameter can be one of the following values:
  - LL_RCC_RTC_NOCLOCK
  - LL_RCC_RTC_HSE_DIV_2
  - LL_RCC_RTC_HSE_DIV_3
  - LL_RCC_RTC_HSE_DIV_4
  - LL_RCC_RTC_HSE_DIV_5
  - LL_RCC_RTC_HSE_DIV_6
  - LL_RCC_RTC_HSE_DIV_7
  - LL_RCC_RTC_HSE_DIV_8
  - LL_RCC_RTC_HSE_DIV_9
  - LL_RCC_RTC_HSE_DIV_10
  - LL_RCC_RTC_HSE_DIV_11
  - LL_RCC_RTC_HSE_DIV_12
  - LL_RCC_RTC_HSE_DIV_13
  - LL_RCC_RTC_HSE_DIV_14
  - LL_RCC_RTC_HSE_DIV_15
  - LL_RCC_RTC_HSE_DIV_16
  - LL_RCC_RTC_HSE_DIV_17
  - LL_RCC_RTC_HSE_DIV_18
  - LL_RCC_RTC_HSE_DIV_19
  - LL_RCC_RTC_HSE_DIV_20
  - LL_RCC_RTC_HSE_DIV_21
  - LL_RCC_RTC_HSE_DIV_22
  - LL_RCC_RTC_HSE_DIV_23
  - LL_RCC_RTC_HSE_DIV_24
  - LL_RCC_RTC_HSE_DIV_25
  - LL_RCC_RTC_HSE_DIV_26
  - LL_RCC_RTC_HSE_DIV_27
  - LL_RCC_RTC_HSE_DIV_28
  - LL_RCC_RTC_HSE_DIV_29
  - LL_RCC_RTC_HSE_DIV_30
  - LL_RCC_RTC_HSE_DIV_31

Return values

- **None:**

Reference Manual to LL API cross reference:

- CFGR RTCPRE LL_RCC_SetRTC_HSEPrescaler

LL_RCC_GetRTC_HSEPrescaler

Function name

__STATIC_INLINE uint32_t LL_RCC_GetRTC_HSEPrescaler (void)

Function description

Get HSE Prescalers for RTC Clock.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_RTC_NOCLOCK
  - LL_RCC_RTC_HSE_DIV_2
  - LL_RCC_RTC_HSE_DIV_3
  - LL_RCC_RTC_HSE_DIV_4
  - LL_RCC_RTC_HSE_DIV_5
  - LL_RCC_RTC_HSE_DIV_6
  - LL_RCC_RTC_HSE_DIV_7
  - LL_RCC_RTC_HSE_DIV_8
  - LL_RCC_RTC_HSE_DIV_9
  - LL_RCC_RTC_HSE_DIV_10
  - LL_RCC_RTC_HSE_DIV_11
  - LL_RCC_RTC_HSE_DIV_12
  - LL_RCC_RTC_HSE_DIV_13
  - LL_RCC_RTC_HSE_DIV_14
  - LL_RCC_RTC_HSE_DIV_15
  - LL_RCC_RTC_HSE_DIV_16
  - LL_RCC_RTC_HSE_DIV_17
  - LL_RCC_RTC_HSE_DIV_18
  - LL_RCC_RTC_HSE_DIV_19
  - LL_RCC_RTC_HSE_DIV_20
  - LL_RCC_RTC_HSE_DIV_21
  - LL_RCC_RTC_HSE_DIV_22
  - LL_RCC_RTC_HSE_DIV_23
  - LL_RCC_RTC_HSE_DIV_24
  - LL_RCC_RTC_HSE_DIV_25
  - LL_RCC_RTC_HSE_DIV_26
  - LL_RCC_RTC_HSE_DIV_27
  - LL_RCC_RTC_HSE_DIV_28
  - LL_RCC_RTC_HSE_DIV_29
  - LL_RCC_RTC_HSE_DIV_30
  - LL_RCC_RTC_HSE_DIV_31

Reference Manual to LL API cross reference:

- CFGR RTCPRE LL_RCC_GetRTC_HSEPrescaler

LL_RCC_SetTIMPrescaler

Function name

__STATIC_INLINE void LL_RCC_SetTIMPrescaler (uint32_t Prescaler)

Function description

Set Timers Clock Prescalers.

Parameters

- **Prescaler:** This parameter can be one of the following values:
  - LL_RCC_TIM_PRESCALER_TWICE
  - LL_RCC_TIM_PRESCALER_FOUR_TIMES

Return values

- **None:**

Reference Manual to LL API cross reference:

- DCKCFGR TIMPRE LL_RCC_SetTIMPrescaler

LL_RCC_GetTIMPrescaler

Function name

__STATIC_INLINE uint32_t LL_RCC_GetTIMPrescaler (void)

Function description

Get Timers Clock Prescalers.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_TIM_PRESCALER_TWICE
  - LL_RCC_TIM_PRESCALER_FOUR_TIMES

Reference Manual to LL API cross reference:

- DCKCFGR TIMPRE LL_RCC_GetTIMPrescaler

LL_RCC_PLL_Enable

Function name

__STATIC_INLINE void LL_RCC_PLL_Enable (void)

Function description

Enable PLL.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PLLON LL_RCC_PLL_Enable

LL_RCC_PLL_Disable

Function name

__STATIC_INLINE void LL_RCC_PLL_Disable (void)

Function description

Disable PLL.

Return values

- **None:**

Notes

- Cannot be disabled if the PLL clock is used as the system clock

Reference Manual to LL API cross reference:

- CR PLLON LL_RCC_PLL_Disable

LL_RCC_PLL_IsReady

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_IsReady (void)

Function description

Check if PLL Ready.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR PLLRDY LL_RCC_PLL_IsReady

LL_RCC_PLL_ConfigDomain_SYS

Function name

__STATIC_INLINE void LL_RCC_PLL_ConfigDomain_SYS (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLP_R)

Function description

Configure PLL used for SYSCLK Domain.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51
  - LL_RCC_PLLM_DIV_52
  - LL_RCC_PLLM_DIV_53

- **PLLN:** Between 50/192(*) and 432
- **PLLP_R:** This parameter can be one of the following values:
  - LL_RCC_PLLP_DIV_2
  - LL_RCC_PLLP_DIV_4
  - LL_RCC_PLLP_DIV_6
  - LL_RCC_PLLP_DIV_8
  - LL_RCC_PLLR_DIV_2 (*)
  - LL_RCC_PLLR_DIV_3 (*)
  - LL_RCC_PLLR_DIV_4 (*)
  - LL_RCC_PLLR_DIV_5 (*)
  - LL_RCC_PLLR_DIV_6 (*)
  - LL_RCC_PLLR_DIV_7 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI(*) are disabled
- PLLN/PLLP can be written only when PLL is disabled

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLL_ConfigDomain_SYS
- PLLCFGR PLLM LL_RCC_PLL_ConfigDomain_SYS
- PLLCFGR PLLN LL_RCC_PLL_ConfigDomain_SYS
- PLLCFGR PLLR LL_RCC_PLL_ConfigDomain_SYS
- PLLCFGR PLLP LL_RCC_PLL_ConfigDomain_SYS

LL_RCC_PLL_ConfigDomain_48M

Function name

__STATIC_INLINE void LL_RCC_PLL_ConfigDomain_48M (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLQ)

Function description

Configure PLL used for 48Mhz domain clock.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51
  - LL_RCC_PLLM_DIV_52
  - LL_RCC_PLLM_DIV_53

- **PLLN:** Between 50/192(*) and 432
- **PLLQ:** This parameter can be one of the following values:
  - LL_RCC_PLLQ_DIV_2
  - LL_RCC_PLLQ_DIV_3
  - LL_RCC_PLLQ_DIV_4
  - LL_RCC_PLLQ_DIV_5
  - LL_RCC_PLLQ_DIV_6
  - LL_RCC_PLLQ_DIV_7
  - LL_RCC_PLLQ_DIV_8
  - LL_RCC_PLLQ_DIV_9
  - LL_RCC_PLLQ_DIV_10
  - LL_RCC_PLLQ_DIV_11
  - LL_RCC_PLLQ_DIV_12
  - LL_RCC_PLLQ_DIV_13
  - LL_RCC_PLLQ_DIV_14
  - LL_RCC_PLLQ_DIV_15

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI(*) are disabled
- PLLN/PLLQ can be written only when PLL is disabled
- This can be selected for USB, RNG, SDIO

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLL_ConfigDomain_48M
- PLLCFGR PLLM LL_RCC_PLL_ConfigDomain_48M
- PLLCFGR PLLN LL_RCC_PLL_ConfigDomain_48M
- PLLCFGR PLLQ LL_RCC_PLL_ConfigDomain_48M

LL_RCC_PLL_ConfigDomain_DSI

Function name

__STATIC_INLINE void LL_RCC_PLL_ConfigDomain_DSI (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLR)

Function description

Configure PLL used for DSI clock.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51
  - LL_RCC_PLLM_DIV_52
  - LL_RCC_PLLM_DIV_53

- **PLLN:** Between 50 and 432
- **PLLR:** This parameter can be one of the following values:
  - LL_RCC_PLLR_DIV_2
  - LL_RCC_PLLR_DIV_3
  - LL_RCC_PLLR_DIV_4
  - LL_RCC_PLLR_DIV_5
  - LL_RCC_PLLR_DIV_6
  - LL_RCC_PLLR_DIV_7

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI are disabled
- PLLN/PLLR can be written only when PLL is disabled
- This can be selected for DSI

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLL_ConfigDomain_DSI
- PLLCFGR PLLM LL_RCC_PLL_ConfigDomain_DSI
- PLLCFGR PLLN LL_RCC_PLL_ConfigDomain_DSI
- PLLCFGR PLLR LL_RCC_PLL_ConfigDomain_DSI

LL_RCC_PLL_ConfigDomain_SAI

Function name

__STATIC_INLINE void LL_RCC_PLL_ConfigDomain_SAI (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLR)

Function description

Configure PLL used for SAI clock.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51
  - LL_RCC_PLLM_DIV_52
  - LL_RCC_PLLM_DIV_53

- **PLLN:** Between 50 and 432
- **PLLR:** This parameter can be one of the following values:
  - LL_RCC_PLLR_DIV_2
  - LL_RCC_PLLR_DIV_3
  - LL_RCC_PLLR_DIV_4
  - LL_RCC_PLLR_DIV_5
  - LL_RCC_PLLR_DIV_6
  - LL_RCC_PLLR_DIV_7
- **PLLDIVR:** This parameter can be one of the following values:
  - LL_RCC_PLLDIVR_DIV_1 (*)
  - LL_RCC_PLLDIVR_DIV_2 (*)
  - LL_RCC_PLLDIVR_DIV_3 (*)
  - LL_RCC_PLLDIVR_DIV_4 (*)
  - LL_RCC_PLLDIVR_DIV_5 (*)
  - LL_RCC_PLLDIVR_DIV_6 (*)
  - LL_RCC_PLLDIVR_DIV_7 (*)
  - LL_RCC_PLLDIVR_DIV_8 (*)
  - LL_RCC_PLLDIVR_DIV_9 (*)
  - LL_RCC_PLLDIVR_DIV_10 (*)
  - LL_RCC_PLLDIVR_DIV_11 (*)
  - LL_RCC_PLLDIVR_DIV_12 (*)
  - LL_RCC_PLLDIVR_DIV_13 (*)
  - LL_RCC_PLLDIVR_DIV_14 (*)
  - LL_RCC_PLLDIVR_DIV_15 (*)
  - LL_RCC_PLLDIVR_DIV_16 (*)
  - LL_RCC_PLLDIVR_DIV_17 (*)
  - LL_RCC_PLLDIVR_DIV_18 (*)
  - LL_RCC_PLLDIVR_DIV_19 (*)
  - LL_RCC_PLLDIVR_DIV_20 (*)
  - LL_RCC_PLLDIVR_DIV_21 (*)
  - LL_RCC_PLLDIVR_DIV_22 (*)
  - LL_RCC_PLLDIVR_DIV_23 (*)
  - LL_RCC_PLLDIVR_DIV_24 (*)
  - LL_RCC_PLLDIVR_DIV_25 (*)
  - LL_RCC_PLLDIVR_DIV_26 (*)
  - LL_RCC_PLLDIVR_DIV_27 (*)
  - LL_RCC_PLLDIVR_DIV_28 (*)
  - LL_RCC_PLLDIVR_DIV_29 (*)
  - LL_RCC_PLLDIVR_DIV_30 (*)
  - LL_RCC_PLLDIVR_DIV_31 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI are disabled
- PLLN/PLLR can be written only when PLL is disabled
- This can be selected for SAI

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLL_ConfigDomain_SAI
- PLLCFGR PLLM LL_RCC_PLL_ConfigDomain_SAI
- PLLCFGR PLLN LL_RCC_PLL_ConfigDomain_SAI
- PLLCFGR PLLR LL_RCC_PLL_ConfigDomain_SAI
- DCKCFGR PLLDIVR LL_RCC_PLL_ConfigDomain_SAI

LL_RCC_PLL_SetMainSource

Function name

__STATIC_INLINE void LL_RCC_PLL_SetMainSource (uint32_t PLLSource)

Function description

Configure PLL clock source.

Parameters

- **PLLSource:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

Return values

- **None:**

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLL_SetMainSource

LL_RCC_PLL_GetMainSource

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetMainSource (void)

Function description

Get the oscillator used as PLL clock source.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLL_GetMainSource

LL_RCC_PLL_GetN

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetN (void)

Function description

Get Main PLL multiplication factor for VCO.

Return values

- **Between:** 50/192(*) and 432

Reference Manual to LL API cross reference:

- PLLCFGR PLLN LL_RCC_PLL_GetN

LL_RCC_PLL_GetP

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetP (void)

Function description

Get Main PLL division factor for PLLP.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLP_DIV_2
  - LL_RCC_PLLP_DIV_4
  - LL_RCC_PLLP_DIV_6
  - LL_RCC_PLLP_DIV_8

Reference Manual to LL API cross reference:

- PLLCFGR PLLP LL_RCC_PLL_GetP

LL_RCC_PLL_GetQ

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetQ (void)

Function description

Get Main PLL division factor for PLLQ.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLQ_DIV_2
  - LL_RCC_PLLQ_DIV_3
  - LL_RCC_PLLQ_DIV_4
  - LL_RCC_PLLQ_DIV_5
  - LL_RCC_PLLQ_DIV_6
  - LL_RCC_PLLQ_DIV_7
  - LL_RCC_PLLQ_DIV_8
  - LL_RCC_PLLQ_DIV_9
  - LL_RCC_PLLQ_DIV_10
  - LL_RCC_PLLQ_DIV_11
  - LL_RCC_PLLQ_DIV_12
  - LL_RCC_PLLQ_DIV_13
  - LL_RCC_PLLQ_DIV_14
  - LL_RCC_PLLQ_DIV_15

Notes

- used for PLL48MCLK selected for USB, RNG, SDIO (48 MHz clock)

Reference Manual to LL API cross reference:

- PLLCFGR PLLQ LL_RCC_PLL_GetQ

LL_RCC_PLL_GetR

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetR (void)

Function description

Get Main PLL division factor for PLLR.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLR_DIV_2
  - LL_RCC_PLLR_DIV_3
  - LL_RCC_PLLR_DIV_4
  - LL_RCC_PLLR_DIV_5
  - LL_RCC_PLLR_DIV_6
  - LL_RCC_PLLR_DIV_7

Notes

- used for PLLCLK (system clock)

Reference Manual to LL API cross reference:

- PLLCFGR PLLR LL_RCC_PLL_GetR

LL_RCC_PLL_GetDivider

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetDivider (void)

Function description

Get Division factor for the main PLL and other PLL.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51
  - LL_RCC_PLLM_DIV_52

Reference Manual to LL API cross reference:

- PLLCFGR PLLM LL_RCC_PLL_GetDivider

LL_RCC_PLL_ConfigSpreadSpectrum

Function name

__STATIC_INLINE void LL_RCC_PLL_ConfigSpreadSpectrum (uint32_t Mod, uint32_t Inc, uint32_t Sel)

Function description

Configure Spread Spectrum used for PLL.

Parameters

- **Mod:** Between Min_Data=0 and Max_Data=8191
- **Inc:** Between Min_Data=0 and Max_Data=32767
- **Sel:** This parameter can be one of the following values:
  - LL_RCC_SPREAD_SELECT_CENTER
  - LL_RCC_SPREAD_SELECT_DOWN

Return values

- **None:**

Notes

- These bits must be written before enabling PLL

Reference Manual to LL API cross reference:

- SSCGR MODPER LL_RCC_PLL_ConfigSpreadSpectrum
- SSCGR INCSTEP LL_RCC_PLL_ConfigSpreadSpectrum
- SSCGR SPREADSEL LL_RCC_PLL_ConfigSpreadSpectrum

LL_RCC_PLL_GetPeriodModulation

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetPeriodModulation (void)

Function description

Get Spread Spectrum Modulation Period for PLL.

Return values

- **Between:** Min_Data=0 and Max_Data=8191

Reference Manual to LL API cross reference:

- SSCGR MODPER LL_RCC_PLL_GetPeriodModulation

LL_RCC_PLL_GetStepIncrementation

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetStepIncrementation (void)

Function description

Get Spread Spectrum Incrementation Step for PLL.

Return values

- **Between:** Min_Data=0 and Max_Data=32767

Notes

- Must be written before enabling PLL

Reference Manual to LL API cross reference:

- SSCGR INCSTEP LL_RCC_PLL_GetStepIncrementation

LL_RCC_PLL_GetSpreadSelection

Function name

__STATIC_INLINE uint32_t LL_RCC_PLL_GetSpreadSelection (void)

Function description

Get Spread Spectrum Selection for PLL.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_SPREAD_SELECT_CENTER
  - LL_RCC_SPREAD_SELECT_DOWN

Notes

- Must be written before enabling PLL

Reference Manual to LL API cross reference:

- SSCGR SPREADSEL LL_RCC_PLL_GetSpreadSelection

LL_RCC_PLL_SpreadSpectrum_Enable

Function name

__STATIC_INLINE void LL_RCC_PLL_SpreadSpectrum_Enable (void)

Function description

Enable Spread Spectrum for PLL.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SSCGR SSCGEN LL_RCC_PLL_SpreadSpectrum_Enable

LL_RCC_PLL_SpreadSpectrum_Disable

Function name

__STATIC_INLINE void LL_RCC_PLL_SpreadSpectrum_Disable (void)

Function description

Disable Spread Spectrum for PLL.

Return values

- **None:**

Reference Manual to LL API cross reference:

- SSCGR SSCGEN LL_RCC_PLL_SpreadSpectrum_Disable

LL_RCC_PLLI2S_Enable

Function name

__STATIC_INLINE void LL_RCC_PLLI2S_Enable (void)

Function description

Enable PLLI2S.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PLLI2SON LL_RCC_PLLI2S_Enable

LL_RCC_PLLI2S_Disable

Function name

__STATIC_INLINE void LL_RCC_PLLI2S_Disable (void)

Function description

Disable PLLI2S.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PLLI2SON LL_RCC_PLLI2S_Disable

LL_RCC_PLLI2S_IsReady

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLI2S_IsReady (void)

Function description

Check if PLLI2S Ready.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR PLLI2SRDY LL_RCC_PLLI2S_IsReady

LL_RCC_PLLI2S_ConfigDomain_SAI

Function name

__STATIC_INLINE void LL_RCC_PLLI2S_ConfigDomain_SAI (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLQ_R, uint32_t PLLDIVQ_R)

Function description

Configure PLLI2S used for SAI domain clock.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE
  - LL_RCC_PLLI2SSOURCE_PIN (*)

  (*) value not defined in all devices.

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLI2SM_DIV_2
  - LL_RCC_PLLI2SM_DIV_3
  - LL_RCC_PLLI2SM_DIV_4
  - LL_RCC_PLLI2SM_DIV_5
  - LL_RCC_PLLI2SM_DIV_6
  - LL_RCC_PLLI2SM_DIV_7
  - LL_RCC_PLLI2SM_DIV_8
  - LL_RCC_PLLI2SM_DIV_9
  - LL_RCC_PLLI2SM_DIV_10
  - LL_RCC_PLLI2SM_DIV_11
  - LL_RCC_PLLI2SM_DIV_12
  - LL_RCC_PLLI2SM_DIV_13
  - LL_RCC_PLLI2SM_DIV_14
  - LL_RCC_PLLI2SM_DIV_15
  - LL_RCC_PLLI2SM_DIV_16
  - LL_RCC_PLLI2SM_DIV_17
  - LL_RCC_PLLI2SM_DIV_18
  - LL_RCC_PLLI2SM_DIV_19
  - LL_RCC_PLLI2SM_DIV_20
  - LL_RCC_PLLI2SM_DIV_21
  - LL_RCC_PLLI2SM_DIV_22
  - LL_RCC_PLLI2SM_DIV_23
  - LL_RCC_PLLI2SM_DIV_24
  - LL_RCC_PLLI2SM_DIV_25
  - LL_RCC_PLLI2SM_DIV_26
  - LL_RCC_PLLI2SM_DIV_27
  - LL_RCC_PLLI2SM_DIV_28
  - LL_RCC_PLLI2SM_DIV_29
  - LL_RCC_PLLI2SM_DIV_30
  - LL_RCC_PLLI2SM_DIV_31
  - LL_RCC_PLLI2SM_DIV_32
  - LL_RCC_PLLI2SM_DIV_33
  - LL_RCC_PLLI2SM_DIV_34
  - LL_RCC_PLLI2SM_DIV_35
  - LL_RCC_PLLI2SM_DIV_36
  - LL_RCC_PLLI2SM_DIV_37
  - LL_RCC_PLLI2SM_DIV_38
  - LL_RCC_PLLI2SM_DIV_39
  - LL_RCC_PLLI2SM_DIV_40
  - LL_RCC_PLLI2SM_DIV_41
  - LL_RCC_PLLI2SM_DIV_42
  - LL_RCC_PLLI2SM_DIV_43
  - LL_RCC_PLLI2SM_DIV_44
  - LL_RCC_PLLI2SM_DIV_45
  - LL_RCC_PLLI2SM_DIV_46
  - LL_RCC_PLLI2SM_DIV_47
  - LL_RCC_PLLI2SM_DIV_48
  - LL_RCC_PLLI2SM_DIV_49
  - LL_RCC_PLLI2SM_DIV_50
  - LL_RCC_PLLI2SM_DIV_51
  - LL_RCC_PLLI2SM_DIV_52
  - LL_RCC_PLLI2SM_DIV_53

- **PLLN:** Between 50/192(*) and 432
- **PLLQ_R:** This parameter can be one of the following values:
  - LL_RCC_PLLI2SQ_DIV_2 (*)
  - LL_RCC_PLLI2SQ_DIV_3 (*)
  - LL_RCC_PLLI2SQ_DIV_4 (*)
  - LL_RCC_PLLI2SQ_DIV_5 (*)
  - LL_RCC_PLLI2SQ_DIV_6 (*)
  - LL_RCC_PLLI2SQ_DIV_7 (*)
  - LL_RCC_PLLI2SQ_DIV_8 (*)
  - LL_RCC_PLLI2SQ_DIV_9 (*)
  - LL_RCC_PLLI2SQ_DIV_10 (*)
  - LL_RCC_PLLI2SQ_DIV_11 (*)
  - LL_RCC_PLLI2SQ_DIV_12 (*)
  - LL_RCC_PLLI2SQ_DIV_13 (*)
  - LL_RCC_PLLI2SQ_DIV_14 (*)
  - LL_RCC_PLLI2SQ_DIV_15 (*)
  - LL_RCC_PLLI2SR_DIV_2 (*)
  - LL_RCC_PLLI2SR_DIV_3 (*)
  - LL_RCC_PLLI2SR_DIV_4 (*)
  - LL_RCC_PLLI2SR_DIV_5 (*)
  - LL_RCC_PLLI2SR_DIV_6 (*)
  - LL_RCC_PLLI2SR_DIV_7 (*)

  (*) value not defined in all devices.

- **PLLDIVQ_R:** This parameter can be one of the following values:
  - LL_RCC_PLLI2SDIVQ_DIV_1 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_2 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_3 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_4 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_5 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_6 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_7 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_8 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_9 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_10 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_11 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_12 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_13 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_14 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_15 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_16 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_17 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_18 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_19 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_20 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_21 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_22 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_23 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_24 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_25 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_26 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_27 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_28 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_29 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_30 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_31 (*)
  - LL_RCC_PLLI2SDIVQ_DIV_32 (*)
  - LL_RCC_PLLI2SDIVR_DIV_1 (*)
  - LL_RCC_PLLI2SDIVR_DIV_2 (*)
  - LL_RCC_PLLI2SDIVR_DIV_3 (*)
  - LL_RCC_PLLI2SDIVR_DIV_4 (*)
  - LL_RCC_PLLI2SDIVR_DIV_5 (*)
  - LL_RCC_PLLI2SDIVR_DIV_6 (*)
  - LL_RCC_PLLI2SDIVR_DIV_7 (*)
  - LL_RCC_PLLI2SDIVR_DIV_8 (*)
  - LL_RCC_PLLI2SDIVR_DIV_9 (*)
  - LL_RCC_PLLI2SDIVR_DIV_10 (*)
  - LL_RCC_PLLI2SDIVR_DIV_11 (*)
  - LL_RCC_PLLI2SDIVR_DIV_12 (*)
  - LL_RCC_PLLI2SDIVR_DIV_13 (*)
  - LL_RCC_PLLI2SDIVR_DIV_14 (*)
  - LL_RCC_PLLI2SDIVR_DIV_15 (*)
  - LL_RCC_PLLI2SDIVR_DIV_16 (*)
  - LL_RCC_PLLI2SDIVR_DIV_17 (*)
  - LL_RCC_PLLI2SDIVR_DIV_18 (*)
  - LL_RCC_PLLI2SDIVR_DIV_19 (*)
  - LL_RCC_PLLI2SDIVR_DIV_20 (*)

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI(*) are disabled
- PLLN/PLLQ/PLLR can be written only when PLLI2S is disabled
- This can be selected for SAI

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLLI2S_ConfigDomain_SAI
- PLLI2SCFGR PLLI2SSRC LL_RCC_PLLI2S_ConfigDomain_SAI
- PLLCFGR PLLM LL_RCC_PLLI2S_ConfigDomain_SAI
- PLLI2SCFGR PLLI2SM LL_RCC_PLLI2S_ConfigDomain_SAI
- PLLI2SCFGR PLLI2SN LL_RCC_PLLI2S_ConfigDomain_SAI
- PLLI2SCFGR PLLI2SQ LL_RCC_PLLI2S_ConfigDomain_SAI
- PLLI2SCFGR PLLI2SR LL_RCC_PLLI2S_ConfigDomain_SAI
- DCKCFGR PLLI2SDIVQ LL_RCC_PLLI2S_ConfigDomain_SAI
- DCKCFGR PLLI2SDIVR LL_RCC_PLLI2S_ConfigDomain_SAI

LL_RCC_PLLI2S_ConfigDomain_I2S

Function name

__STATIC_INLINE void LL_RCC_PLLI2S_ConfigDomain_I2S (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLR)

Function description

Configure PLLI2S used for I2S1 domain clock.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE
  - LL_RCC_PLLI2SSOURCE_PIN (*)

  (*) value not defined in all devices.

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLI2SM_DIV_2
  - LL_RCC_PLLI2SM_DIV_3
  - LL_RCC_PLLI2SM_DIV_4
  - LL_RCC_PLLI2SM_DIV_5
  - LL_RCC_PLLI2SM_DIV_6
  - LL_RCC_PLLI2SM_DIV_7
  - LL_RCC_PLLI2SM_DIV_8
  - LL_RCC_PLLI2SM_DIV_9
  - LL_RCC_PLLI2SM_DIV_10
  - LL_RCC_PLLI2SM_DIV_11
  - LL_RCC_PLLI2SM_DIV_12
  - LL_RCC_PLLI2SM_DIV_13
  - LL_RCC_PLLI2SM_DIV_14
  - LL_RCC_PLLI2SM_DIV_15
  - LL_RCC_PLLI2SM_DIV_16
  - LL_RCC_PLLI2SM_DIV_17
  - LL_RCC_PLLI2SM_DIV_18
  - LL_RCC_PLLI2SM_DIV_19
  - LL_RCC_PLLI2SM_DIV_20
  - LL_RCC_PLLI2SM_DIV_21
  - LL_RCC_PLLI2SM_DIV_22
  - LL_RCC_PLLI2SM_DIV_23
  - LL_RCC_PLLI2SM_DIV_24
  - LL_RCC_PLLI2SM_DIV_25
  - LL_RCC_PLLI2SM_DIV_26
  - LL_RCC_PLLI2SM_DIV_27
  - LL_RCC_PLLI2SM_DIV_28
  - LL_RCC_PLLI2SM_DIV_29
  - LL_RCC_PLLI2SM_DIV_30
  - LL_RCC_PLLI2SM_DIV_31
  - LL_RCC_PLLI2SM_DIV_32
  - LL_RCC_PLLI2SM_DIV_33
  - LL_RCC_PLLI2SM_DIV_34
  - LL_RCC_PLLI2SM_DIV_35
  - LL_RCC_PLLI2SM_DIV_36
  - LL_RCC_PLLI2SM_DIV_37
  - LL_RCC_PLLI2SM_DIV_38
  - LL_RCC_PLLI2SM_DIV_39
  - LL_RCC_PLLI2SM_DIV_40
  - LL_RCC_PLLI2SM_DIV_41
  - LL_RCC_PLLI2SM_DIV_42
  - LL_RCC_PLLI2SM_DIV_43
  - LL_RCC_PLLI2SM_DIV_44
  - LL_RCC_PLLI2SM_DIV_45
  - LL_RCC_PLLI2SM_DIV_46
  - LL_RCC_PLLI2SM_DIV_47
  - LL_RCC_PLLI2SM_DIV_48
  - LL_RCC_PLLI2SM_DIV_49
  - LL_RCC_PLLI2SM_DIV_50
  - LL_RCC_PLLI2SM_DIV_51
  - LL_RCC_PLLI2SM_DIV_52
  - LL_RCC_PLLI2SM_DIV_53

- **PLLN:** Between 50/192(*) and 432
- **PLLR:** This parameter can be one of the following values:
  - LL_RCC_PLLI2SR_DIV_2
  - LL_RCC_PLLI2SR_DIV_3
  - LL_RCC_PLLI2SR_DIV_4
  - LL_RCC_PLLI2SR_DIV_5
  - LL_RCC_PLLI2SR_DIV_6
  - LL_RCC_PLLI2SR_DIV_7

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI(*) are disabled
- PLLN/PLLR can be written only when PLLI2S is disabled
- This can be selected for I2S

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLLI2S_ConfigDomain_I2S
- PLLCFGR PLLM LL_RCC_PLLI2S_ConfigDomain_I2S
- PLLI2SCFGR PLLI2SSRC LL_RCC_PLLI2S_ConfigDomain_I2S
- PLLI2SCFGR PLLI2SM LL_RCC_PLLI2S_ConfigDomain_I2S
- PLLI2SCFGR PLLI2SN LL_RCC_PLLI2S_ConfigDomain_I2S
- PLLI2SCFGR PLLI2SR LL_RCC_PLLI2S_ConfigDomain_I2S

LL_RCC_PLLI2S_GetN

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLI2S_GetN (void)

Function description

Get I2SPLL multiplication factor for VCO.

Return values

- **Between:** 50/192(*) and 432

Reference Manual to LL API cross reference:

- PLLI2SCFGR PLLI2SN LL_RCC_PLLI2S_GetN

LL_RCC_PLLI2S_GetQ

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLI2S_GetQ (void)

Function description

Get I2SPLL division factor for PLLI2SQ.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLI2SQ_DIV_2
  - LL_RCC_PLLI2SQ_DIV_3
  - LL_RCC_PLLI2SQ_DIV_4
  - LL_RCC_PLLI2SQ_DIV_5
  - LL_RCC_PLLI2SQ_DIV_6
  - LL_RCC_PLLI2SQ_DIV_7
  - LL_RCC_PLLI2SQ_DIV_8
  - LL_RCC_PLLI2SQ_DIV_9
  - LL_RCC_PLLI2SQ_DIV_10
  - LL_RCC_PLLI2SQ_DIV_11
  - LL_RCC_PLLI2SQ_DIV_12
  - LL_RCC_PLLI2SQ_DIV_13
  - LL_RCC_PLLI2SQ_DIV_14
  - LL_RCC_PLLI2SQ_DIV_15

Reference Manual to LL API cross reference:

- PLLI2SCFGR PLLI2SQ LL_RCC_PLLI2S_GetQ

LL_RCC_PLLI2S_GetR

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLI2S_GetR (void)

Function description

Get I2SPLL division factor for PLLI2SR.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLI2SR_DIV_2
  - LL_RCC_PLLI2SR_DIV_3
  - LL_RCC_PLLI2SR_DIV_4
  - LL_RCC_PLLI2SR_DIV_5
  - LL_RCC_PLLI2SR_DIV_6
  - LL_RCC_PLLI2SR_DIV_7

Notes

- used for PLLI2SCLK (I2S clock)

Reference Manual to LL API cross reference:

- PLLI2SCFGR PLLI2SR LL_RCC_PLLI2S_GetR

LL_RCC_PLLI2S_GetDIVQ

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLI2S_GetDIVQ (void)

Function description

Get I2SPLL division factor for PLLI2SDIVQ.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLI2SDIVQ_DIV_1
  - LL_RCC_PLLI2SDIVQ_DIV_2
  - LL_RCC_PLLI2SDIVQ_DIV_3
  - LL_RCC_PLLI2SDIVQ_DIV_4
  - LL_RCC_PLLI2SDIVQ_DIV_5
  - LL_RCC_PLLI2SDIVQ_DIV_6
  - LL_RCC_PLLI2SDIVQ_DIV_7
  - LL_RCC_PLLI2SDIVQ_DIV_8
  - LL_RCC_PLLI2SDIVQ_DIV_9
  - LL_RCC_PLLI2SDIVQ_DIV_10
  - LL_RCC_PLLI2SDIVQ_DIV_11
  - LL_RCC_PLLI2SDIVQ_DIV_12
  - LL_RCC_PLLI2SDIVQ_DIV_13
  - LL_RCC_PLLI2SDIVQ_DIV_14
  - LL_RCC_PLLI2SDIVQ_DIV_15
  - LL_RCC_PLLI2SDIVQ_DIV_16
  - LL_RCC_PLLI2SDIVQ_DIV_17
  - LL_RCC_PLLI2SDIVQ_DIV_18
  - LL_RCC_PLLI2SDIVQ_DIV_19
  - LL_RCC_PLLI2SDIVQ_DIV_20
  - LL_RCC_PLLI2SDIVQ_DIV_21
  - LL_RCC_PLLI2SDIVQ_DIV_22
  - LL_RCC_PLLI2SDIVQ_DIV_23
  - LL_RCC_PLLI2SDIVQ_DIV_24
  - LL_RCC_PLLI2SDIVQ_DIV_25
  - LL_RCC_PLLI2SDIVQ_DIV_26
  - LL_RCC_PLLI2SDIVQ_DIV_27
  - LL_RCC_PLLI2SDIVQ_DIV_28
  - LL_RCC_PLLI2SDIVQ_DIV_29
  - LL_RCC_PLLI2SDIVQ_DIV_30
  - LL_RCC_PLLI2SDIVQ_DIV_31
  - LL_RCC_PLLI2SDIVQ_DIV_32

Notes

- used PLLSAICLK selected (SAI clock)

Reference Manual to LL API cross reference:

- DCKCFGR PLLI2SDIVQ LL_RCC_PLLI2S_GetDIVQ

LL_RCC_PLLI2S_GetDivider

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLI2S_GetDivider (void)

Function description

Get division factor for PLLI2S input clock.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLI2SM_DIV_2
  - LL_RCC_PLLI2SM_DIV_3
  - LL_RCC_PLLI2SM_DIV_4
  - LL_RCC_PLLI2SM_DIV_5
  - LL_RCC_PLLI2SM_DIV_6
  - LL_RCC_PLLI2SM_DIV_7
  - LL_RCC_PLLI2SM_DIV_8
  - LL_RCC_PLLI2SM_DIV_9
  - LL_RCC_PLLI2SM_DIV_10
  - LL_RCC_PLLI2SM_DIV_11
  - LL_RCC_PLLI2SM_DIV_12
  - LL_RCC_PLLI2SM_DIV_13
  - LL_RCC_PLLI2SM_DIV_14
  - LL_RCC_PLLI2SM_DIV_15
  - LL_RCC_PLLI2SM_DIV_16
  - LL_RCC_PLLI2SM_DIV_17
  - LL_RCC_PLLI2SM_DIV_18
  - LL_RCC_PLLI2SM_DIV_19
  - LL_RCC_PLLI2SM_DIV_20
  - LL_RCC_PLLI2SM_DIV_21
  - LL_RCC_PLLI2SM_DIV_22
  - LL_RCC_PLLI2SM_DIV_23
  - LL_RCC_PLLI2SM_DIV_24
  - LL_RCC_PLLI2SM_DIV_25
  - LL_RCC_PLLI2SM_DIV_26
  - LL_RCC_PLLI2SM_DIV_27
  - LL_RCC_PLLI2SM_DIV_28
  - LL_RCC_PLLI2SM_DIV_29
  - LL_RCC_PLLI2SM_DIV_30
  - LL_RCC_PLLI2SM_DIV_31
  - LL_RCC_PLLI2SM_DIV_32
  - LL_RCC_PLLI2SM_DIV_33
  - LL_RCC_PLLI2SM_DIV_34
  - LL_RCC_PLLI2SM_DIV_35
  - LL_RCC_PLLI2SM_DIV_36
  - LL_RCC_PLLI2SM_DIV_37
  - LL_RCC_PLLI2SM_DIV_38
  - LL_RCC_PLLI2SM_DIV_39
  - LL_RCC_PLLI2SM_DIV_40
  - LL_RCC_PLLI2SM_DIV_41
  - LL_RCC_PLLI2SM_DIV_42
  - LL_RCC_PLLI2SM_DIV_43
  - LL_RCC_PLLI2SM_DIV_44
  - LL_RCC_PLLI2SM_DIV_45
  - LL_RCC_PLLI2SM_DIV_46
  - LL_RCC_PLLI2SM_DIV_47
  - LL_RCC_PLLI2SM_DIV_48
  - LL_RCC_PLLI2SM_DIV_49
  - LL_RCC_PLLI2SM_DIV_50
  - LL_RCC_PLLI2SM_DIV_51
  - LL_RCC_PLLI2SM_DIV_52

Reference Manual to LL API cross reference:

- PLLCFGR PLLM LL_RCC_PLLI2S_GetDivider
- PLLI2SCFGR PLLI2SM LL_RCC_PLLI2S_GetDivider

LL_RCC_PLLI2S_GetMainSource

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLI2S_GetMainSource (void)

Function description

Get the oscillator used as PLL clock source.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE
  - LL_RCC_PLLI2SSOURCE_PIN (*)

  (*) value not defined in all devices.

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLLI2S_GetMainSource
- PLLI2SCFGR PLLI2SSRC LL_RCC_PLLI2S_GetMainSource

LL_RCC_PLLSAI_Enable

Function name

__STATIC_INLINE void LL_RCC_PLLSAI_Enable (void)

Function description

Enable PLLSAI.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PLLSAION LL_RCC_PLLSAI_Enable

LL_RCC_PLLSAI_Disable

Function name

__STATIC_INLINE void LL_RCC_PLLSAI_Disable (void)

Function description

Disable PLLSAI.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR PLLSAION LL_RCC_PLLSAI_Disable

LL_RCC_PLLSAI_IsReady

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLSAI_IsReady (void)

Function description

Check if PLLSAI Ready.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR PLLSAIRDY LL_RCC_PLLSAI_IsReady

LL_RCC_PLLSAI_ConfigDomain_SAI

Function name

__STATIC_INLINE void LL_RCC_PLLSAI_ConfigDomain_SAI (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLQ, uint32_t PLLDIVQ)

Function description

Configure PLLSAI used for SAI domain clock.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLSAIM_DIV_2
  - LL_RCC_PLLSAIM_DIV_3
  - LL_RCC_PLLSAIM_DIV_4
  - LL_RCC_PLLSAIM_DIV_5
  - LL_RCC_PLLSAIM_DIV_6
  - LL_RCC_PLLSAIM_DIV_7
  - LL_RCC_PLLSAIM_DIV_8
  - LL_RCC_PLLSAIM_DIV_9
  - LL_RCC_PLLSAIM_DIV_10
  - LL_RCC_PLLSAIM_DIV_11
  - LL_RCC_PLLSAIM_DIV_12
  - LL_RCC_PLLSAIM_DIV_13
  - LL_RCC_PLLSAIM_DIV_14
  - LL_RCC_PLLSAIM_DIV_15
  - LL_RCC_PLLSAIM_DIV_16
  - LL_RCC_PLLSAIM_DIV_17
  - LL_RCC_PLLSAIM_DIV_18
  - LL_RCC_PLLSAIM_DIV_19
  - LL_RCC_PLLSAIM_DIV_20
  - LL_RCC_PLLSAIM_DIV_21
  - LL_RCC_PLLSAIM_DIV_22
  - LL_RCC_PLLSAIM_DIV_23
  - LL_RCC_PLLSAIM_DIV_24
  - LL_RCC_PLLSAIM_DIV_25
  - LL_RCC_PLLSAIM_DIV_26
  - LL_RCC_PLLSAIM_DIV_27
  - LL_RCC_PLLSAIM_DIV_28
  - LL_RCC_PLLSAIM_DIV_29
  - LL_RCC_PLLSAIM_DIV_30
  - LL_RCC_PLLSAIM_DIV_31
  - LL_RCC_PLLSAIM_DIV_32
  - LL_RCC_PLLSAIM_DIV_33
  - LL_RCC_PLLSAIM_DIV_34
  - LL_RCC_PLLSAIM_DIV_35
  - LL_RCC_PLLSAIM_DIV_36
  - LL_RCC_PLLSAIM_DIV_37
  - LL_RCC_PLLSAIM_DIV_38
  - LL_RCC_PLLSAIM_DIV_39
  - LL_RCC_PLLSAIM_DIV_40
  - LL_RCC_PLLSAIM_DIV_41
  - LL_RCC_PLLSAIM_DIV_42
  - LL_RCC_PLLSAIM_DIV_43
  - LL_RCC_PLLSAIM_DIV_44
  - LL_RCC_PLLSAIM_DIV_45
  - LL_RCC_PLLSAIM_DIV_46
  - LL_RCC_PLLSAIM_DIV_47
  - LL_RCC_PLLSAIM_DIV_48
  - LL_RCC_PLLSAIM_DIV_49
  - LL_RCC_PLLSAIM_DIV_50
  - LL_RCC_PLLSAIM_DIV_51
  - LL_RCC_PLLSAIM_DIV_52
  - LL_RCC_PLLSAIM_DIV_53

- **PLLN:** Between 49/50(*) and 432
- **PLLQ:** This parameter can be one of the following values:
  - LL_RCC_PLLSAIQ_DIV_2
  - LL_RCC_PLLSAIQ_DIV_3
  - LL_RCC_PLLSAIQ_DIV_4
  - LL_RCC_PLLSAIQ_DIV_5
  - LL_RCC_PLLSAIQ_DIV_6
  - LL_RCC_PLLSAIQ_DIV_7
  - LL_RCC_PLLSAIQ_DIV_8
  - LL_RCC_PLLSAIQ_DIV_9
  - LL_RCC_PLLSAIQ_DIV_10
  - LL_RCC_PLLSAIQ_DIV_11
  - LL_RCC_PLLSAIQ_DIV_12
  - LL_RCC_PLLSAIQ_DIV_13
  - LL_RCC_PLLSAIQ_DIV_14
  - LL_RCC_PLLSAIQ_DIV_15
- **PLLDIVQ:** This parameter can be one of the following values:
  - LL_RCC_PLLSAIDIVQ_DIV_1
  - LL_RCC_PLLSAIDIVQ_DIV_2
  - LL_RCC_PLLSAIDIVQ_DIV_3
  - LL_RCC_PLLSAIDIVQ_DIV_4
  - LL_RCC_PLLSAIDIVQ_DIV_5
  - LL_RCC_PLLSAIDIVQ_DIV_6
  - LL_RCC_PLLSAIDIVQ_DIV_7
  - LL_RCC_PLLSAIDIVQ_DIV_8
  - LL_RCC_PLLSAIDIVQ_DIV_9
  - LL_RCC_PLLSAIDIVQ_DIV_10
  - LL_RCC_PLLSAIDIVQ_DIV_11
  - LL_RCC_PLLSAIDIVQ_DIV_12
  - LL_RCC_PLLSAIDIVQ_DIV_13
  - LL_RCC_PLLSAIDIVQ_DIV_14
  - LL_RCC_PLLSAIDIVQ_DIV_15
  - LL_RCC_PLLSAIDIVQ_DIV_16
  - LL_RCC_PLLSAIDIVQ_DIV_17
  - LL_RCC_PLLSAIDIVQ_DIV_18
  - LL_RCC_PLLSAIDIVQ_DIV_19
  - LL_RCC_PLLSAIDIVQ_DIV_20
  - LL_RCC_PLLSAIDIVQ_DIV_21
  - LL_RCC_PLLSAIDIVQ_DIV_22
  - LL_RCC_PLLSAIDIVQ_DIV_23
  - LL_RCC_PLLSAIDIVQ_DIV_24
  - LL_RCC_PLLSAIDIVQ_DIV_25
  - LL_RCC_PLLSAIDIVQ_DIV_26
  - LL_RCC_PLLSAIDIVQ_DIV_27
  - LL_RCC_PLLSAIDIVQ_DIV_28
  - LL_RCC_PLLSAIDIVQ_DIV_29
  - LL_RCC_PLLSAIDIVQ_DIV_30
  - LL_RCC_PLLSAIDIVQ_DIV_31
  - LL_RCC_PLLSAIDIVQ_DIV_32

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI(*) are disabled
- PLLN/PLLQ can be written only when PLLSAI is disabled
- This can be selected for SAI

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLLSAI_ConfigDomain_SAI
- PLLCFGR PLLM LL_RCC_PLLSAI_ConfigDomain_SAI
- PLLSAICFGR PLLSAIM LL_RCC_PLLSAI_ConfigDomain_SAI
- PLLSAICFGR PLLSAIN LL_RCC_PLLSAI_ConfigDomain_SAI
- PLLSAICFGR PLLSAIQ LL_RCC_PLLSAI_ConfigDomain_SAI
- DCKCFGR PLLSAIDIVQ LL_RCC_PLLSAI_ConfigDomain_SAI

LL_RCC_PLLSAI_ConfigDomain_48M

Function name

__STATIC_INLINE void LL_RCC_PLLSAI_ConfigDomain_48M (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLP)

Function description

Configure PLLSAI used for 48Mhz domain clock.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLSAIM_DIV_2
  - LL_RCC_PLLSAIM_DIV_3
  - LL_RCC_PLLSAIM_DIV_4
  - LL_RCC_PLLSAIM_DIV_5
  - LL_RCC_PLLSAIM_DIV_6
  - LL_RCC_PLLSAIM_DIV_7
  - LL_RCC_PLLSAIM_DIV_8
  - LL_RCC_PLLSAIM_DIV_9
  - LL_RCC_PLLSAIM_DIV_10
  - LL_RCC_PLLSAIM_DIV_11
  - LL_RCC_PLLSAIM_DIV_12
  - LL_RCC_PLLSAIM_DIV_13
  - LL_RCC_PLLSAIM_DIV_14
  - LL_RCC_PLLSAIM_DIV_15
  - LL_RCC_PLLSAIM_DIV_16
  - LL_RCC_PLLSAIM_DIV_17
  - LL_RCC_PLLSAIM_DIV_18
  - LL_RCC_PLLSAIM_DIV_19
  - LL_RCC_PLLSAIM_DIV_20
  - LL_RCC_PLLSAIM_DIV_21
  - LL_RCC_PLLSAIM_DIV_22
  - LL_RCC_PLLSAIM_DIV_23
  - LL_RCC_PLLSAIM_DIV_24
  - LL_RCC_PLLSAIM_DIV_25
  - LL_RCC_PLLSAIM_DIV_26
  - LL_RCC_PLLSAIM_DIV_27
  - LL_RCC_PLLSAIM_DIV_28
  - LL_RCC_PLLSAIM_DIV_29
  - LL_RCC_PLLSAIM_DIV_30
  - LL_RCC_PLLSAIM_DIV_31
  - LL_RCC_PLLSAIM_DIV_32
  - LL_RCC_PLLSAIM_DIV_33
  - LL_RCC_PLLSAIM_DIV_34
  - LL_RCC_PLLSAIM_DIV_35
  - LL_RCC_PLLSAIM_DIV_36
  - LL_RCC_PLLSAIM_DIV_37
  - LL_RCC_PLLSAIM_DIV_38
  - LL_RCC_PLLSAIM_DIV_39
  - LL_RCC_PLLSAIM_DIV_40
  - LL_RCC_PLLSAIM_DIV_41
  - LL_RCC_PLLSAIM_DIV_42
  - LL_RCC_PLLSAIM_DIV_43
  - LL_RCC_PLLSAIM_DIV_44
  - LL_RCC_PLLSAIM_DIV_45
  - LL_RCC_PLLSAIM_DIV_46
  - LL_RCC_PLLSAIM_DIV_47
  - LL_RCC_PLLSAIM_DIV_48
  - LL_RCC_PLLSAIM_DIV_49
  - LL_RCC_PLLSAIM_DIV_50
  - LL_RCC_PLLSAIM_DIV_51
  - LL_RCC_PLLSAIM_DIV_52
  - LL_RCC_PLLSAIM_DIV_53

- **PLLN:** Between 50 and 432
- **PLLP:** This parameter can be one of the following values:
  - LL_RCC_PLLSAIP_DIV_2
  - LL_RCC_PLLSAIP_DIV_4
  - LL_RCC_PLLSAIP_DIV_6
  - LL_RCC_PLLSAIP_DIV_8

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI(*) are disabled
- PLLN/PLLP can be written only when PLLSAI is disabled
- This can be selected for USB, RNG, SDIO

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLLSAI_ConfigDomain_48M
- PLLCFGR PLLM LL_RCC_PLLSAI_ConfigDomain_48M
- PLLSAICFGR PLLSAIM LL_RCC_PLLSAI_ConfigDomain_48M
- PLLSAICFGR PLLSAIN LL_RCC_PLLSAI_ConfigDomain_48M
- PLLSAICFGR PLLSAIP LL_RCC_PLLSAI_ConfigDomain_48M

LL_RCC_PLLSAI_ConfigDomain_LTDC

Function name

__STATIC_INLINE void LL_RCC_PLLSAI_ConfigDomain_LTDC (uint32_t Source, uint32_t PLLM, uint32_t PLLN, uint32_t PLLR, uint32_t PLLDIVR)

Function description

Configure PLLSAI used for LTDC domain clock.

Parameters

- **Source:** This parameter can be one of the following values:
  - LL_RCC_PLLSOURCE_HSI
  - LL_RCC_PLLSOURCE_HSE

- **PLLM:** This parameter can be one of the following values:
  - LL_RCC_PLLSAIM_DIV_2
  - LL_RCC_PLLSAIM_DIV_3
  - LL_RCC_PLLSAIM_DIV_4
  - LL_RCC_PLLSAIM_DIV_5
  - LL_RCC_PLLSAIM_DIV_6
  - LL_RCC_PLLSAIM_DIV_7
  - LL_RCC_PLLSAIM_DIV_8
  - LL_RCC_PLLSAIM_DIV_9
  - LL_RCC_PLLSAIM_DIV_10
  - LL_RCC_PLLSAIM_DIV_11
  - LL_RCC_PLLSAIM_DIV_12
  - LL_RCC_PLLSAIM_DIV_13
  - LL_RCC_PLLSAIM_DIV_14
  - LL_RCC_PLLSAIM_DIV_15
  - LL_RCC_PLLSAIM_DIV_16
  - LL_RCC_PLLSAIM_DIV_17
  - LL_RCC_PLLSAIM_DIV_18
  - LL_RCC_PLLSAIM_DIV_19
  - LL_RCC_PLLSAIM_DIV_20
  - LL_RCC_PLLSAIM_DIV_21
  - LL_RCC_PLLSAIM_DIV_22
  - LL_RCC_PLLSAIM_DIV_23
  - LL_RCC_PLLSAIM_DIV_24
  - LL_RCC_PLLSAIM_DIV_25
  - LL_RCC_PLLSAIM_DIV_26
  - LL_RCC_PLLSAIM_DIV_27
  - LL_RCC_PLLSAIM_DIV_28
  - LL_RCC_PLLSAIM_DIV_29
  - LL_RCC_PLLSAIM_DIV_30
  - LL_RCC_PLLSAIM_DIV_31
  - LL_RCC_PLLSAIM_DIV_32
  - LL_RCC_PLLSAIM_DIV_33
  - LL_RCC_PLLSAIM_DIV_34
  - LL_RCC_PLLSAIM_DIV_35
  - LL_RCC_PLLSAIM_DIV_36
  - LL_RCC_PLLSAIM_DIV_37
  - LL_RCC_PLLSAIM_DIV_38
  - LL_RCC_PLLSAIM_DIV_39
  - LL_RCC_PLLSAIM_DIV_40
  - LL_RCC_PLLSAIM_DIV_41
  - LL_RCC_PLLSAIM_DIV_42
  - LL_RCC_PLLSAIM_DIV_43
  - LL_RCC_PLLSAIM_DIV_44
  - LL_RCC_PLLSAIM_DIV_45
  - LL_RCC_PLLSAIM_DIV_46
  - LL_RCC_PLLSAIM_DIV_47
  - LL_RCC_PLLSAIM_DIV_48
  - LL_RCC_PLLSAIM_DIV_49
  - LL_RCC_PLLSAIM_DIV_50
  - LL_RCC_PLLSAIM_DIV_51
  - LL_RCC_PLLSAIM_DIV_52
  - LL_RCC_PLLSAIM_DIV_53

- **PLLN:** Between 49/50(*) and 432
- **PLLR:** This parameter can be one of the following values:
  - LL_RCC_PLLSAIR_DIV_2
  - LL_RCC_PLLSAIR_DIV_3
  - LL_RCC_PLLSAIR_DIV_4
  - LL_RCC_PLLSAIR_DIV_5
  - LL_RCC_PLLSAIR_DIV_6
  - LL_RCC_PLLSAIR_DIV_7
- **PLLDIVR:** This parameter can be one of the following values:
  - LL_RCC_PLLSAIDIVR_DIV_2
  - LL_RCC_PLLSAIDIVR_DIV_4
  - LL_RCC_PLLSAIDIVR_DIV_8
  - LL_RCC_PLLSAIDIVR_DIV_16

Return values

- **None:**

Notes

- PLL Source and PLLM Divider can be written only when PLL, PLLI2S and PLLSAI(*) are disabled
- PLLN/PLLR can be written only when PLLSAI is disabled
- This can be selected for LTDC

Reference Manual to LL API cross reference:

- PLLCFGR PLLSRC LL_RCC_PLLSAI_ConfigDomain_LTDC
- PLLCFGR PLLM LL_RCC_PLLSAI_ConfigDomain_LTDC
- PLLSAICFGR PLLSAIN LL_RCC_PLLSAI_ConfigDomain_LTDC
- PLLSAICFGR PLLSAIR LL_RCC_PLLSAI_ConfigDomain_LTDC
- DCKCFGR PLLSAIDIVR LL_RCC_PLLSAI_ConfigDomain_LTDC

LL_RCC_PLLSAI_GetDivider

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLSAI_GetDivider (void)

Function description

Get division factor for PLLSAI input clock.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLSAIM_DIV_2
  - LL_RCC_PLLSAIM_DIV_3
  - LL_RCC_PLLSAIM_DIV_4
  - LL_RCC_PLLSAIM_DIV_5
  - LL_RCC_PLLSAIM_DIV_6
  - LL_RCC_PLLSAIM_DIV_7
  - LL_RCC_PLLSAIM_DIV_8
  - LL_RCC_PLLSAIM_DIV_9
  - LL_RCC_PLLSAIM_DIV_10
  - LL_RCC_PLLSAIM_DIV_11
  - LL_RCC_PLLSAIM_DIV_12
  - LL_RCC_PLLSAIM_DIV_13
  - LL_RCC_PLLSAIM_DIV_14
  - LL_RCC_PLLSAIM_DIV_15
  - LL_RCC_PLLSAIM_DIV_16
  - LL_RCC_PLLSAIM_DIV_17
  - LL_RCC_PLLSAIM_DIV_18
  - LL_RCC_PLLSAIM_DIV_19
  - LL_RCC_PLLSAIM_DIV_20
  - LL_RCC_PLLSAIM_DIV_21
  - LL_RCC_PLLSAIM_DIV_22
  - LL_RCC_PLLSAIM_DIV_23
  - LL_RCC_PLLSAIM_DIV_24
  - LL_RCC_PLLSAIM_DIV_25
  - LL_RCC_PLLSAIM_DIV_26
  - LL_RCC_PLLSAIM_DIV_27
  - LL_RCC_PLLSAIM_DIV_28
  - LL_RCC_PLLSAIM_DIV_29
  - LL_RCC_PLLSAIM_DIV_30
  - LL_RCC_PLLSAIM_DIV_31
  - LL_RCC_PLLSAIM_DIV_32
  - LL_RCC_PLLSAIM_DIV_33
  - LL_RCC_PLLSAIM_DIV_34
  - LL_RCC_PLLSAIM_DIV_35
  - LL_RCC_PLLSAIM_DIV_36
  - LL_RCC_PLLSAIM_DIV_37
  - LL_RCC_PLLSAIM_DIV_38
  - LL_RCC_PLLSAIM_DIV_39
  - LL_RCC_PLLSAIM_DIV_40
  - LL_RCC_PLLSAIM_DIV_41
  - LL_RCC_PLLSAIM_DIV_42
  - LL_RCC_PLLSAIM_DIV_43
  - LL_RCC_PLLSAIM_DIV_44
  - LL_RCC_PLLSAIM_DIV_45
  - LL_RCC_PLLSAIM_DIV_46
  - LL_RCC_PLLSAIM_DIV_47
  - LL_RCC_PLLSAIM_DIV_48
  - LL_RCC_PLLSAIM_DIV_49
  - LL_RCC_PLLSAIM_DIV_50
  - LL_RCC_PLLSAIM_DIV_51
  - LL_RCC_PLLSAIM_DIV_52

Reference Manual to LL API cross reference:

- PLLCFGR PLLM LL_RCC_PLLSAI_GetDivider
- PLLSAICFGR PLLSAIM LL_RCC_PLLSAI_GetDivider

LL_RCC_PLLSAI_GetN

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLSAI_GetN (void)

Function description

Get SAIPLL multiplication factor for VCO.

Return values

- **Between:** 49/50(*) and 432

Reference Manual to LL API cross reference:

- PLLSAICFGR PLLSAIN LL_RCC_PLLSAI_GetN

LL_RCC_PLLSAI_GetQ

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLSAI_GetQ (void)

Function description

Get SAIPLL division factor for PLLSAIQ.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLSAIQ_DIV_2
  - LL_RCC_PLLSAIQ_DIV_3
  - LL_RCC_PLLSAIQ_DIV_4
  - LL_RCC_PLLSAIQ_DIV_5
  - LL_RCC_PLLSAIQ_DIV_6
  - LL_RCC_PLLSAIQ_DIV_7
  - LL_RCC_PLLSAIQ_DIV_8
  - LL_RCC_PLLSAIQ_DIV_9
  - LL_RCC_PLLSAIQ_DIV_10
  - LL_RCC_PLLSAIQ_DIV_11
  - LL_RCC_PLLSAIQ_DIV_12
  - LL_RCC_PLLSAIQ_DIV_13
  - LL_RCC_PLLSAIQ_DIV_14
  - LL_RCC_PLLSAIQ_DIV_15

Reference Manual to LL API cross reference:

- PLLSAICFGR PLLSAIQ LL_RCC_PLLSAI_GetQ

LL_RCC_PLLSAI_GetR

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLSAI_GetR (void)

Function description

Get SAIPLL division factor for PLLSAIR.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLSAIR_DIV_2
  - LL_RCC_PLLSAIR_DIV_3
  - LL_RCC_PLLSAIR_DIV_4
  - LL_RCC_PLLSAIR_DIV_5
  - LL_RCC_PLLSAIR_DIV_6
  - LL_RCC_PLLSAIR_DIV_7

Notes

- used for PLLSAICLK (SAI clock)

Reference Manual to LL API cross reference:

- PLLSAICFGR PLLSAIR LL_RCC_PLLSAI_GetR

LL_RCC_PLLSAI_GetP

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLSAI_GetP (void)

Function description

Get SAIPLL division factor for PLLSAIP.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLSAIP_DIV_2
  - LL_RCC_PLLSAIP_DIV_4
  - LL_RCC_PLLSAIP_DIV_6
  - LL_RCC_PLLSAIP_DIV_8

Notes

- used for PLL48MCLK (48M domain clock)

Reference Manual to LL API cross reference:

- PLLSAICFGR PLLSAIP LL_RCC_PLLSAI_GetP

LL_RCC_PLLSAI_GetDIVQ

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLSAI_GetDIVQ (void)

Function description

Get SAIPLL division factor for PLLSAIDIVQ.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLSAIDIVQ_DIV_1
  - LL_RCC_PLLSAIDIVQ_DIV_2
  - LL_RCC_PLLSAIDIVQ_DIV_3
  - LL_RCC_PLLSAIDIVQ_DIV_4
  - LL_RCC_PLLSAIDIVQ_DIV_5
  - LL_RCC_PLLSAIDIVQ_DIV_6
  - LL_RCC_PLLSAIDIVQ_DIV_7
  - LL_RCC_PLLSAIDIVQ_DIV_8
  - LL_RCC_PLLSAIDIVQ_DIV_9
  - LL_RCC_PLLSAIDIVQ_DIV_10
  - LL_RCC_PLLSAIDIVQ_DIV_11
  - LL_RCC_PLLSAIDIVQ_DIV_12
  - LL_RCC_PLLSAIDIVQ_DIV_13
  - LL_RCC_PLLSAIDIVQ_DIV_14
  - LL_RCC_PLLSAIDIVQ_DIV_15
  - LL_RCC_PLLSAIDIVQ_DIV_16
  - LL_RCC_PLLSAIDIVQ_DIV_17
  - LL_RCC_PLLSAIDIVQ_DIV_18
  - LL_RCC_PLLSAIDIVQ_DIV_19
  - LL_RCC_PLLSAIDIVQ_DIV_20
  - LL_RCC_PLLSAIDIVQ_DIV_21
  - LL_RCC_PLLSAIDIVQ_DIV_22
  - LL_RCC_PLLSAIDIVQ_DIV_23
  - LL_RCC_PLLSAIDIVQ_DIV_24
  - LL_RCC_PLLSAIDIVQ_DIV_25
  - LL_RCC_PLLSAIDIVQ_DIV_26
  - LL_RCC_PLLSAIDIVQ_DIV_27
  - LL_RCC_PLLSAIDIVQ_DIV_28
  - LL_RCC_PLLSAIDIVQ_DIV_29
  - LL_RCC_PLLSAIDIVQ_DIV_30
  - LL_RCC_PLLSAIDIVQ_DIV_31
  - LL_RCC_PLLSAIDIVQ_DIV_32

Notes

- used PLLSAICLK selected (SAI clock)

Reference Manual to LL API cross reference:

- DCKCFGR PLLSAIDIVQ LL_RCC_PLLSAI_GetDIVQ

LL_RCC_PLLSAI_GetDIVR

Function name

__STATIC_INLINE uint32_t LL_RCC_PLLSAI_GetDIVR (void)

Function description

Get SAIPLL division factor for PLLSAIDIVR.

Return values

- **Returned:** value can be one of the following values:
  - LL_RCC_PLLSAIDIVR_DIV_2
  - LL_RCC_PLLSAIDIVR_DIV_4
  - LL_RCC_PLLSAIDIVR_DIV_8
  - LL_RCC_PLLSAIDIVR_DIV_16

Notes

- used for LTDC domain clock

Reference Manual to LL API cross reference:

- DCKCFGR PLLSAIDIVR LL_RCC_PLLSAI_GetDIVR

LL_RCC_ClearFlag_LSIRDY

Function name

__STATIC_INLINE void LL_RCC_ClearFlag_LSIRDY (void)

Function description

Clear LSI ready interrupt flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR LSIRDYC LL_RCC_ClearFlag_LSIRDY

LL_RCC_ClearFlag_LSERDY

Function name

__STATIC_INLINE void LL_RCC_ClearFlag_LSERDY (void)

Function description

Clear LSE ready interrupt flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR LSERDYC LL_RCC_ClearFlag_LSERDY

LL_RCC_ClearFlag_HSIRDY

Function name

__STATIC_INLINE void LL_RCC_ClearFlag_HSIRDY (void)

Function description

Clear HSI ready interrupt flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR HSIRDYC LL_RCC_ClearFlag_HSIRDY

LL_RCC_ClearFlag_HSERDY

Function name

__STATIC_INLINE void LL_RCC_ClearFlag_HSERDY (void)

Function description

Clear HSE ready interrupt flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR HSERDYC LL_RCC_ClearFlag_HSERDY

LL_RCC_ClearFlag_PLLRDY

Function name

__STATIC_INLINE void LL_RCC_ClearFlag_PLLRDY (void)

Function description

Clear PLL ready interrupt flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLRDYC LL_RCC_ClearFlag_PLLRDY

LL_RCC_ClearFlag_PLLI2SRDY

Function name

__STATIC_INLINE void LL_RCC_ClearFlag_PLLI2SRDY (void)

Function description

Clear PLLI2S ready interrupt flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLI2SRDYC LL_RCC_ClearFlag_PLLI2SRDY

LL_RCC_ClearFlag_PLLSAIRDY

Function name

__STATIC_INLINE void LL_RCC_ClearFlag_PLLSAIRDY (void)

Function description

Clear PLLSAI ready interrupt flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLSAIRDYC LL_RCC_ClearFlag_PLLSAIRDY

LL_RCC_ClearFlag_HSECSS

Function name

__STATIC_INLINE void LL_RCC_ClearFlag_HSECSS (void)

Function description

Clear Clock security system interrupt flag.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR CSSC LL_RCC_ClearFlag_HSECSS

LL_RCC_IsActiveFlag_LSIRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LSIRDY (void)

Function description

Check if LSI ready interrupt occurred or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR LSIRDYF LL_RCC_IsActiveFlag_LSIRDY

LL_RCC_IsActiveFlag_LSERDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LSERDY (void)

Function description

Check if LSE ready interrupt occurred or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR LSERDYF LL_RCC_IsActiveFlag_LSERDY

LL_RCC_IsActiveFlag_HSIRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSIRDY (void)

Function description

Check if HSI ready interrupt occurred or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR HSIRDYF LL_RCC_IsActiveFlag_HSIRDY

LL_RCC_IsActiveFlag_HSERDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSERDY (void)

Function description

Check if HSE ready interrupt occurred or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR HSERDYF LL_RCC_IsActiveFlag_HSERDY

LL_RCC_IsActiveFlag_PLLRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PLLRDY (void)

Function description

Check if PLL ready interrupt occurred or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR PLLRDYF LL_RCC_IsActiveFlag_PLLRDY

LL_RCC_IsActiveFlag_PLLI2SRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PLLI2SRDY (void)

Function description

Check if PLLI2S ready interrupt occurred or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR PLLI2SRDYF LL_RCC_IsActiveFlag_PLLI2SRDY

LL_RCC_IsActiveFlag_PLLSAIRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PLLSAIRDY (void)

Function description

Check if PLLSAI ready interrupt occurred or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR PLLSAIRDYF LL_RCC_IsActiveFlag_PLLSAIRDY

LL_RCC_IsActiveFlag_HSECSS

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_HSECSS (void)

Function description

Check if Clock security system interrupt occurred or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR CSSF LL_RCC_IsActiveFlag_HSECSS

LL_RCC_IsActiveFlag_IWDGRST

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_IWDGRST (void)

Function description

Check if RCC flag Independent Watchdog reset is set or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR IWDGRSTF LL_RCC_IsActiveFlag_IWDGRST

LL_RCC_IsActiveFlag_LPWRRST

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_LPWRRST (void)

Function description

Check if RCC flag Low Power reset is set or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR LPWRRSTF LL_RCC_IsActiveFlag_LPWRRST

LL_RCC_IsActiveFlag_PINRST

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PINRST (void)

Function description

Check if RCC flag Pin reset is set or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR PINRSTF LL_RCC_IsActiveFlag_PINRST

LL_RCC_IsActiveFlag_PORRST

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_PORRST (void)

Function description

Check if RCC flag POR/PDR reset is set or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR PORRSTF LL_RCC_IsActiveFlag_PORRST

LL_RCC_IsActiveFlag_SFTRST

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_SFTRST (void)

Function description

Check if RCC flag Software reset is set or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR SFTRSTF LL_RCC_IsActiveFlag_SFTRST

LL_RCC_IsActiveFlag_WWDGRST

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_WWDGRST (void)

Function description

Check if RCC flag Window Watchdog reset is set or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR WWDGRSTF LL_RCC_IsActiveFlag_WWDGRST

LL_RCC_IsActiveFlag_BORRST

Function name

__STATIC_INLINE uint32_t LL_RCC_IsActiveFlag_BORRST (void)

Function description

Check if RCC flag BOR reset is set or not.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CSR BORRSTF LL_RCC_IsActiveFlag_BORRST

LL_RCC_ClearResetFlags

Function name

__STATIC_INLINE void LL_RCC_ClearResetFlags (void)

Function description

Set RMVF bit to clear the reset flags.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CSR RMVF LL_RCC_ClearResetFlags

LL_RCC_EnableIT_LSIRDY

Function name

__STATIC_INLINE void LL_RCC_EnableIT_LSIRDY (void)

Function description

Enable LSI ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR LSIRDYIE LL_RCC_EnableIT_LSIRDY

LL_RCC_EnableIT_LSERDY

Function name

__STATIC_INLINE void LL_RCC_EnableIT_LSERDY (void)

Function description

Enable LSE ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR LSERDYIE LL_RCC_EnableIT_LSERDY

LL_RCC_EnableIT_HSIRDY

Function name

__STATIC_INLINE void LL_RCC_EnableIT_HSIRDY (void)

Function description

Enable HSI ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR HSIRDYIE LL_RCC_EnableIT_HSIRDY

LL_RCC_EnableIT_HSERDY

Function name

__STATIC_INLINE void LL_RCC_EnableIT_HSERDY (void)

Function description

Enable HSE ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR HSERDYIE LL_RCC_EnableIT_HSERDY

LL_RCC_EnableIT_PLLRDY

Function name

__STATIC_INLINE void LL_RCC_EnableIT_PLLRDY (void)

Function description

Enable PLL ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLRDYIE LL_RCC_EnableIT_PLLRDY

LL_RCC_EnableIT_PLLI2SRDY

Function name

__STATIC_INLINE void LL_RCC_EnableIT_PLLI2SRDY (void)

Function description

Enable PLLI2S ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLI2SRDYIE LL_RCC_EnableIT_PLLI2SRDY

LL_RCC_EnableIT_PLLSAIRDY

Function name

__STATIC_INLINE void LL_RCC_EnableIT_PLLSAIRDY (void)

Function description

Enable PLLSAI ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLSAIRDYIE LL_RCC_EnableIT_PLLSAIRDY

LL_RCC_DisableIT_LSIRDY

Function name

__STATIC_INLINE void LL_RCC_DisableIT_LSIRDY (void)

Function description

Disable LSI ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR LSIRDYIE LL_RCC_DisableIT_LSIRDY

LL_RCC_DisableIT_LSERDY

Function name

__STATIC_INLINE void LL_RCC_DisableIT_LSERDY (void)

Function description

Disable LSE ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR LSERDYIE LL_RCC_DisableIT_LSERDY

LL_RCC_DisableIT_HSIRDY

Function name

__STATIC_INLINE void LL_RCC_DisableIT_HSIRDY (void)

Function description

Disable HSI ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR HSIRDYIE LL_RCC_DisableIT_HSIRDY

LL_RCC_DisableIT_HSERDY

Function name

__STATIC_INLINE void LL_RCC_DisableIT_HSERDY (void)

Function description

Disable HSE ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR HSERDYIE LL_RCC_DisableIT_HSERDY

LL_RCC_DisableIT_PLLRDY

Function name

__STATIC_INLINE void LL_RCC_DisableIT_PLLRDY (void)

Function description

Disable PLL ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLRDYIE LL_RCC_DisableIT_PLLRDY

LL_RCC_DisableIT_PLLI2SRDY

Function name

__STATIC_INLINE void LL_RCC_DisableIT_PLLI2SRDY (void)

Function description

Disable PLLI2S ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLI2SRDYIE LL_RCC_DisableIT_PLLI2SRDY

LL_RCC_DisableIT_PLLSAIRDY

Function name

__STATIC_INLINE void LL_RCC_DisableIT_PLLSAIRDY (void)

Function description

Disable PLLSAI ready interrupt.

Return values

- **None:**

Reference Manual to LL API cross reference:

- CIR PLLSAIRDYIE LL_RCC_DisableIT_PLLSAIRDY

LL_RCC_IsEnabledIT_LSIRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_LSIRDY (void)

Function description

Checks if LSI ready interrupt source is enabled or disabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR LSIRDYIE LL_RCC_IsEnabledIT_LSIRDY

LL_RCC_IsEnabledIT_LSERDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_LSERDY (void)

Function description

Checks if LSE ready interrupt source is enabled or disabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR LSERDYIE LL_RCC_IsEnabledIT_LSERDY

LL_RCC_IsEnabledIT_HSIRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_HSIRDY (void)

Function description

Checks if HSI ready interrupt source is enabled or disabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR HSIRDYIE LL_RCC_IsEnabledIT_HSIRDY

LL_RCC_IsEnabledIT_HSERDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_HSERDY (void)

Function description

Checks if HSE ready interrupt source is enabled or disabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR HSERDYIE LL_RCC_IsEnabledIT_HSERDY

LL_RCC_IsEnabledIT_PLLRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_PLLRDY (void)

Function description

Checks if PLL ready interrupt source is enabled or disabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR PLLRDYIE LL_RCC_IsEnabledIT_PLLRDY

LL_RCC_IsEnabledIT_PLLI2SRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_PLLI2SRDY (void)

Function description

Checks if PLLI2S ready interrupt source is enabled or disabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR PLLI2SRDYIE LL_RCC_IsEnabledIT_PLLI2SRDY

LL_RCC_IsEnabledIT_PLLSAIRDY

Function name

__STATIC_INLINE uint32_t LL_RCC_IsEnabledIT_PLLSAIRDY (void)

Function description

Checks if PLLSAI ready interrupt source is enabled or disabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CIR PLLSAIRDYIE LL_RCC_IsEnabledIT_PLLSAIRDY

LL_RCC_DeInit

Function name

ErrorStatus LL_RCC_DeInit (void)

Function description

Reset the RCC clock configuration to the default reset state.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RCC registers are de-initialized
  - ERROR: not applicable

Notes

- The default reset state of the clock configuration is given below: HSI ON and used as system clock sourceHSE and PLL OFFAHB, APB1 and APB2 prescaler set to 1.CSS, MCO OFFAll interrupts disabled
- This function doesn't modify the configuration of the Peripheral clocksLSI, LSE and RTC clocks

LL_RCC_GetSystemClocksFreq

Function name

void LL_RCC_GetSystemClocksFreq (LL_RCC_ClocksTypeDef * RCC_Clocks)

Function description

Return the frequencies of different on chip clocks; System, AHB, APB1 and APB2 buses clocks.

Parameters

- **RCC_Clocks:** pointer to a LL_RCC_ClocksTypeDef structure which will hold the clocks frequencies

Return values

- **None:**

Notes

- Each time SYSCLK, HCLK, PCLK1 and/or PCLK2 clock changes, this function must be called to update structure fields. Otherwise, any configuration based on this function will be incorrect.

LL_RCC_GetSAIClockFreq

Function name

uint32_t LL_RCC_GetSAIClockFreq (uint32_t SAIxSource)

Function description

Return SAIx clock frequency.

Parameters

- **SAIxSource:** This parameter can be one of the following values:
  - LL_RCC_SAI1_CLKSOURCE (*)
  - LL_RCC_SAI2_CLKSOURCE (*)
  - LL_RCC_SAI1_A_CLKSOURCE (*)
  - LL_RCC_SAI1_B_CLKSOURCE (*)

  (*) value not defined in all devices.

Return values

- **SAI:** clock frequency (in Hz)
  - LL_RCC_PERIPH_FREQUENCY_NO indicates that oscillator is not ready

LL_RCC_GetSDIOClockFreq

Function name

uint32_t LL_RCC_GetSDIOClockFreq (uint32_t SDIOxSource)

Function description

Return SDIOx clock frequency.

Parameters

- **SDIOxSource:** This parameter can be one of the following values:
  - LL_RCC_SDIO_CLKSOURCE

Return values

- **SDIO:** clock frequency (in Hz)
  - LL_RCC_PERIPH_FREQUENCY_NO indicates that oscillator is not ready

LL_RCC_GetRNGClockFreq

Function name

uint32_t LL_RCC_GetRNGClockFreq (uint32_t RNGxSource)

Function description

Return RNGx clock frequency.

Parameters

- **RNGxSource:** This parameter can be one of the following values:
  - LL_RCC_RNG_CLKSOURCE

Return values

- **RNG:** clock frequency (in Hz)
  - LL_RCC_PERIPH_FREQUENCY_NO indicates that oscillator is not ready

LL_RCC_GetUSBClockFreq

Function name

uint32_t LL_RCC_GetUSBClockFreq (uint32_t USBxSource)

Function description

Return USBx clock frequency.

Parameters

- **USBxSource:** This parameter can be one of the following values:
  - LL_RCC_USB_CLKSOURCE

Return values

- **USB:** clock frequency (in Hz)
  - LL_RCC_PERIPH_FREQUENCY_NO indicates that oscillator is not ready

LL_RCC_GetI2SClockFreq

Function name

uint32_t LL_RCC_GetI2SClockFreq (uint32_t I2SxSource)

Function description

Return I2Sx clock frequency.

Parameters

- **I2SxSource:** This parameter can be one of the following values:
  - LL_RCC_I2S1_CLKSOURCE
  - LL_RCC_I2S2_CLKSOURCE (*)

  (*) value not defined in all devices.

Return values

- **I2S:** clock frequency (in Hz)
  - LL_RCC_PERIPH_FREQUENCY_NO indicates that oscillator is not ready

LL_RCC_GetLTDCClockFreq

Function name

uint32_t LL_RCC_GetLTDCClockFreq (uint32_t LTDCxSource)

Function description

Return LTDC clock frequency.

Parameters

- **LTDCxSource:** This parameter can be one of the following values:
  - LL_RCC_LTDC_CLKSOURCE

Return values

- **LTDC:** clock frequency (in Hz)
  - LL_RCC_PERIPH_FREQUENCY_NO indicates that oscillator PLLSAI is not ready

LL_RCC_GetDSIClockFreq

Function name

uint32_t LL_RCC_GetDSIClockFreq (uint32_t DSIxSource)

Function description

Return DSI clock frequency.

Parameters

- **DSIxSource:** This parameter can be one of the following values:
  - LL_RCC_DSI_CLKSOURCE

Return values

- **DSI:** clock frequency (in Hz)
  - LL_RCC_PERIPH_FREQUENCY_NO indicates that oscillator is not ready
  - LL_RCC_PERIPH_FREQUENCY_NA indicates that external clock is used

87.3 RCC Firmware driver defines

The following section lists the various define and macros of the module.

87.3.1 RCC

RCC

APB low-speed prescaler (APB1)

LL_RCC_APB1_DIV_1

HCLK not divided

LL_RCC_APB1_DIV_2

HCLK divided by 2

LL_RCC_APB1_DIV_4

HCLK divided by 4

LL_RCC_APB1_DIV_8

HCLK divided by 8

LL_RCC_APB1_DIV_16

HCLK divided by 16

APB high-speed prescaler (APB2)

LL_RCC_APB2_DIV_1

HCLK not divided

LL_RCC_APB2_DIV_2

HCLK divided by 2

LL_RCC_APB2_DIV_4

HCLK divided by 4

LL_RCC_APB2_DIV_8

HCLK divided by 8

LL_RCC_APB2_DIV_16

HCLK divided by 16

Peripheral CK48M get clock source

LL_RCC_CK48M_CLKSOURCE

CK48M Domain clock source selection

Peripheral 48Mhz domain clock source selection

LL_RCC_CK48M_CLKSOURCE_PLL

PLL oscillator clock used as 48Mhz domain clock

LL_RCC_CK48M_CLKSOURCE_PLLSAI

PLLSAI oscillator clock used as 48Mhz domain clock

Clear Flags Defines

LL_RCC_CIR_LSIRDYC

LSI Ready Interrupt Clear

LL_RCC_CIR_LSERDYC

LSE Ready Interrupt Clear

LL_RCC_CIR_HSIRDYC

HSI Ready Interrupt Clear

LL_RCC_CIR_HSERDYC

HSE Ready Interrupt Clear

LL_RCC_CIR_PLLRDYC

PLL Ready Interrupt Clear

LL_RCC_CIR_PLLI2SRDYC

PLLI2S Ready Interrupt Clear

LL_RCC_CIR_PLLSAIRDYC

PLLSAI Ready Interrupt Clear

LL_RCC_CIR_CSSC

Clock Security System Interrupt Clear

Peripheral DSI get clock source

LL_RCC_DSI_CLKSOURCE

DSI Clock source selection

Peripheral DSI clock source selection

LL_RCC_DSI_CLKSOURCE_PHY

DSI-PHY clock used as DSI byte lane clock source

LL_RCC_DSI_CLKSOURCE_PLL

PLL clock used as DSI byte lane clock source

Get Flags Defines

LL_RCC_CIR_LSIRDYF

LSI Ready Interrupt flag

LL_RCC_CIR_LSERDYF

LSE Ready Interrupt flag

LL_RCC_CIR_HSIRDYF

HSI Ready Interrupt flag

LL_RCC_CIR_HSERDYF

HSE Ready Interrupt flag

LL_RCC_CIR_PLLRDYF

PLL Ready Interrupt flag

LL_RCC_CIR_PLLI2SRDYF

PLLI2S Ready Interrupt flag

LL_RCC_CIR_PLLSAIRDYF

PLLSAI Ready Interrupt flag

LL_RCC_CIR_CSSF

Clock Security System Interrupt flag

LL_RCC_CSR_LPWRRSTF

Low-Power reset flag

LL_RCC_CSR_PINRSTF

PIN reset flag

LL_RCC_CSR_PORRSTF

POR/PDR reset flag

LL_RCC_CSR_SFTRSTF

Software Reset flag

LL_RCC_CSR_IWDGRSTF

Independent Watchdog reset flag

LL_RCC_CSR_WWDGRSTF

Window watchdog reset flag

LL_RCC_CSR_BORRSTF

BOR reset flag

Peripheral I2S get clock source

LL_RCC_I2S1_CLKSOURCE

I2S1 Clock source selection

Peripheral I2S clock source selection

LL_RCC_I2S1_CLKSOURCE_PLLI2S

I2S oscillator clock used as I2S1 clock

LL_RCC_I2S1_CLKSOURCE_PIN

External pin clock used as I2S1 clock

IT Defines

LL_RCC_CIR_LSIRDYIE

LSI Ready Interrupt Enable

LL_RCC_CIR_LSERDYIE

LSE Ready Interrupt Enable

LL_RCC_CIR_HSIRDYIE

HSI Ready Interrupt Enable

LL_RCC_CIR_HSERDYIE

HSE Ready Interrupt Enable

LL_RCC_CIR_PLLRDYIE

PLL Ready Interrupt Enable

LL_RCC_CIR_PLLI2SRDYIE

PLLI2S Ready Interrupt Enable

LL_RCC_CIR_PLLSAIRDYIE

PLLSAI Ready Interrupt Enable

Peripheral LTDC get clock source

LL_RCC_LTDC_CLKSOURCE

LTDC Clock source selection

MCO source selection

LL_RCC_MCO1SOURCE_HSI

HSI selection as MCO1 source

LL_RCC_MCO1SOURCE_LSE

LSE selection as MCO1 source

LL_RCC_MCO1SOURCE_HSE

HSE selection as MCO1 source

LL_RCC_MCO1SOURCE_PLLCLK

PLLCLK selection as MCO1 source

LL_RCC_MCO2SOURCE_SYSCLK

SYSCLK selection as MCO2 source

LL_RCC_MCO2SOURCE_PLLI2S

PLLI2S selection as MCO2 source

LL_RCC_MCO2SOURCE_HSE

HSE selection as MCO2 source

LL_RCC_MCO2SOURCE_PLLCLK

PLLCLK selection as MCO2 source

MCO prescaler

LL_RCC_MCO1_DIV_1

MCO1 not divided

LL_RCC_MCO1_DIV_2

MCO1 divided by 2

LL_RCC_MCO1_DIV_3

MCO1 divided by 3

LL_RCC_MCO1_DIV_4

MCO1 divided by 4

LL_RCC_MCO1_DIV_5

MCO1 divided by 5

LL_RCC_MCO2_DIV_1

MCO2 not divided

LL_RCC_MCO2_DIV_2

MCO2 divided by 2

LL_RCC_MCO2_DIV_3

MCO2 divided by 3

LL_RCC_MCO2_DIV_4

MCO2 divided by 4

LL_RCC_MCO2_DIV_5

MCO2 divided by 5

Oscillator Values adaptation

HSE_VALUE

Value of the HSE oscillator in Hz

HSI_VALUE

Value of the HSI oscillator in Hz

LSE_VALUE

Value of the LSE oscillator in Hz

LSI_VALUE

Value of the LSI oscillator in Hz

EXTERNAL_CLOCK_VALUE

Value of the I2S_CKIN external oscillator in Hz

Peripheral clock frequency

LL_RCC_PERIPH_FREQUENCY_NO

No clock enabled for the peripheral

LL_RCC_PERIPH_FREQUENCY_NA

Frequency cannot be provided as external clock

PLLI2SDIVQ division factor (PLLI2SDIVQ)

LL_RCC_PLLI2SDIVQ_DIV_1

PLLI2S division factor for PLLI2SDIVQ output by 1

LL_RCC_PLLI2SDIVQ_DIV_2

PLLI2S division factor for PLLI2SDIVQ output by 2

LL_RCC_PLLI2SDIVQ_DIV_3

PLLI2S division factor for PLLI2SDIVQ output by 3

LL_RCC_PLLI2SDIVQ_DIV_4

PLLI2S division factor for PLLI2SDIVQ output by 4

LL_RCC_PLLI2SDIVQ_DIV_5

PLLI2S division factor for PLLI2SDIVQ output by 5

LL_RCC_PLLI2SDIVQ_DIV_6

PLLI2S division factor for PLLI2SDIVQ output by 6

LL_RCC_PLLI2SDIVQ_DIV_7

PLLI2S division factor for PLLI2SDIVQ output by 7

LL_RCC_PLLI2SDIVQ_DIV_8

PLLI2S division factor for PLLI2SDIVQ output by 8

LL_RCC_PLLI2SDIVQ_DIV_9

PLLI2S division factor for PLLI2SDIVQ output by 9

LL_RCC_PLLI2SDIVQ_DIV_10

PLLI2S division factor for PLLI2SDIVQ output by 10

LL_RCC_PLLI2SDIVQ_DIV_11

PLLI2S division factor for PLLI2SDIVQ output by 11

LL_RCC_PLLI2SDIVQ_DIV_12

PLLI2S division factor for PLLI2SDIVQ output by 12

LL_RCC_PLLI2SDIVQ_DIV_13

PLLI2S division factor for PLLI2SDIVQ output by 13

LL_RCC_PLLI2SDIVQ_DIV_14

PLLI2S division factor for PLLI2SDIVQ output by 14

LL_RCC_PLLI2SDIVQ_DIV_15

PLLI2S division factor for PLLI2SDIVQ output by 15

LL_RCC_PLLI2SDIVQ_DIV_16

PLLI2S division factor for PLLI2SDIVQ output by 16

LL_RCC_PLLI2SDIVQ_DIV_17

PLLI2S division factor for PLLI2SDIVQ output by 17

LL_RCC_PLLI2SDIVQ_DIV_18

PLLI2S division factor for PLLI2SDIVQ output by 18

LL_RCC_PLLI2SDIVQ_DIV_19

PLLI2S division factor for PLLI2SDIVQ output by 19

LL_RCC_PLLI2SDIVQ_DIV_20

PLLI2S division factor for PLLI2SDIVQ output by 20

LL_RCC_PLLI2SDIVQ_DIV_21

PLLI2S division factor for PLLI2SDIVQ output by 21

LL_RCC_PLLI2SDIVQ_DIV_22

PLLI2S division factor for PLLI2SDIVQ output by 22

LL_RCC_PLLI2SDIVQ_DIV_23

PLLI2S division factor for PLLI2SDIVQ output by 23

LL_RCC_PLLI2SDIVQ_DIV_24

PLLI2S division factor for PLLI2SDIVQ output by 24

LL_RCC_PLLI2SDIVQ_DIV_25

PLLI2S division factor for PLLI2SDIVQ output by 25

LL_RCC_PLLI2SDIVQ_DIV_26

PLLI2S division factor for PLLI2SDIVQ output by 26

LL_RCC_PLLI2SDIVQ_DIV_27

PLLI2S division factor for PLLI2SDIVQ output by 27

LL_RCC_PLLI2SDIVQ_DIV_28

PLLI2S division factor for PLLI2SDIVQ output by 28

LL_RCC_PLLI2SDIVQ_DIV_29

PLLI2S division factor for PLLI2SDIVQ output by 29

LL_RCC_PLLI2SDIVQ_DIV_30

PLLI2S division factor for PLLI2SDIVQ output by 30

LL_RCC_PLLI2SDIVQ_DIV_31

PLLI2S division factor for PLLI2SDIVQ output by 31

LL_RCC_PLLI2SDIVQ_DIV_32

PLLI2S division factor for PLLI2SDIVQ output by 32

PLLI2SM division factor (PLLI2SM)

LL_RCC_PLLI2SM_DIV_2

PLLI2S division factor for PLLI2SM output by 2

LL_RCC_PLLI2SM_DIV_3

PLLI2S division factor for PLLI2SM output by 3

LL_RCC_PLLI2SM_DIV_4

PLLI2S division factor for PLLI2SM output by 4

LL_RCC_PLLI2SM_DIV_5

PLLI2S division factor for PLLI2SM output by 5

LL_RCC_PLLI2SM_DIV_6

PLLI2S division factor for PLLI2SM output by 6

LL_RCC_PLLI2SM_DIV_7

PLLI2S division factor for PLLI2SM output by 7

LL_RCC_PLLI2SM_DIV_8

PLLI2S division factor for PLLI2SM output by 8

LL_RCC_PLLI2SM_DIV_9

PLLI2S division factor for PLLI2SM output by 9

LL_RCC_PLLI2SM_DIV_10

PLLI2S division factor for PLLI2SM output by 10

LL_RCC_PLLI2SM_DIV_11

PLLI2S division factor for PLLI2SM output by 11

LL_RCC_PLLI2SM_DIV_12

PLLI2S division factor for PLLI2SM output by 12

LL_RCC_PLLI2SM_DIV_13

PLLI2S division factor for PLLI2SM output by 13

LL_RCC_PLLI2SM_DIV_14

PLLI2S division factor for PLLI2SM output by 14

LL_RCC_PLLI2SM_DIV_15

PLLI2S division factor for PLLI2SM output by 15

LL_RCC_PLLI2SM_DIV_16

PLLI2S division factor for PLLI2SM output by 16

LL_RCC_PLLI2SM_DIV_17

PLLI2S division factor for PLLI2SM output by 17

LL_RCC_PLLI2SM_DIV_18

PLLI2S division factor for PLLI2SM output by 18

LL_RCC_PLLI2SM_DIV_19

PLLI2S division factor for PLLI2SM output by 19

LL_RCC_PLLI2SM_DIV_20

PLLI2S division factor for PLLI2SM output by 20

LL_RCC_PLLI2SM_DIV_21

PLLI2S division factor for PLLI2SM output by 21

LL_RCC_PLLI2SM_DIV_22

PLLI2S division factor for PLLI2SM output by 22

LL_RCC_PLLI2SM_DIV_23

PLLI2S division factor for PLLI2SM output by 23

LL_RCC_PLLI2SM_DIV_24

PLLI2S division factor for PLLI2SM output by 24

LL_RCC_PLLI2SM_DIV_25

PLLI2S division factor for PLLI2SM output by 25

LL_RCC_PLLI2SM_DIV_26

PLLI2S division factor for PLLI2SM output by 26

LL_RCC_PLLI2SM_DIV_27

PLLI2S division factor for PLLI2SM output by 27

LL_RCC_PLLI2SM_DIV_28

PLLI2S division factor for PLLI2SM output by 28

LL_RCC_PLLI2SM_DIV_29

PLLI2S division factor for PLLI2SM output by 29

LL_RCC_PLLI2SM_DIV_30

PLLI2S division factor for PLLI2SM output by 30

LL_RCC_PLLI2SM_DIV_31

PLLI2S division factor for PLLI2SM output by 31

LL_RCC_PLLI2SM_DIV_32

PLLI2S division factor for PLLI2SM output by 32

LL_RCC_PLLI2SM_DIV_33

PLLI2S division factor for PLLI2SM output by 33

LL_RCC_PLLI2SM_DIV_34

PLLI2S division factor for PLLI2SM output by 34

LL_RCC_PLLI2SM_DIV_35

PLLI2S division factor for PLLI2SM output by 35

LL_RCC_PLLI2SM_DIV_36

PLLI2S division factor for PLLI2SM output by 36

LL_RCC_PLLI2SM_DIV_37

PLLI2S division factor for PLLI2SM output by 37

LL_RCC_PLLI2SM_DIV_38

PLLI2S division factor for PLLI2SM output by 38

LL_RCC_PLLI2SM_DIV_39

PLLI2S division factor for PLLI2SM output by 39

LL_RCC_PLLI2SM_DIV_40

PLLI2S division factor for PLLI2SM output by 40

LL_RCC_PLLI2SM_DIV_41

PLLI2S division factor for PLLI2SM output by 41

LL_RCC_PLLI2SM_DIV_42

PLLI2S division factor for PLLI2SM output by 42

LL_RCC_PLLI2SM_DIV_43

PLLI2S division factor for PLLI2SM output by 43

LL_RCC_PLLI2SM_DIV_44

PLLI2S division factor for PLLI2SM output by 44

LL_RCC_PLLI2SM_DIV_45

PLLI2S division factor for PLLI2SM output by 45

LL_RCC_PLLI2SM_DIV_46

PLLI2S division factor for PLLI2SM output by 46

LL_RCC_PLLI2SM_DIV_47

PLLI2S division factor for PLLI2SM output by 47

LL_RCC_PLLI2SM_DIV_48

PLLI2S division factor for PLLI2SM output by 48

LL_RCC_PLLI2SM_DIV_49

PLLI2S division factor for PLLI2SM output by 49

LL_RCC_PLLI2SM_DIV_50

PLLI2S division factor for PLLI2SM output by 50

LL_RCC_PLLI2SM_DIV_51

PLLI2S division factor for PLLI2SM output by 51

LL_RCC_PLLI2SM_DIV_52

PLLI2S division factor for PLLI2SM output by 52

LL_RCC_PLLI2SM_DIV_53

PLLI2S division factor for PLLI2SM output by 53

LL_RCC_PLLI2SM_DIV_54

PLLI2S division factor for PLLI2SM output by 54

LL_RCC_PLLI2SM_DIV_55

PLLI2S division factor for PLLI2SM output by 55

LL_RCC_PLLI2SM_DIV_56

PLLI2S division factor for PLLI2SM output by 56

LL_RCC_PLLI2SM_DIV_57

PLLI2S division factor for PLLI2SM output by 57

LL_RCC_PLLI2SM_DIV_58

PLLI2S division factor for PLLI2SM output by 58

LL_RCC_PLLI2SM_DIV_59

PLLI2S division factor for PLLI2SM output by 59

LL_RCC_PLLI2SM_DIV_60

PLLI2S division factor for PLLI2SM output by 60

LL_RCC_PLLI2SM_DIV_61

PLLI2S division factor for PLLI2SM output by 61

LL_RCC_PLLI2SM_DIV_62

PLLI2S division factor for PLLI2SM output by 62

LL_RCC_PLLI2SM_DIV_63

PLLI2S division factor for PLLI2SM output by 63

PLLI2SQ division factor (PLLI2SQ)

LL_RCC_PLLI2SQ_DIV_2

PLLI2S division factor for PLLI2SQ output by 2

LL_RCC_PLLI2SQ_DIV_3

PLLI2S division factor for PLLI2SQ output by 3

LL_RCC_PLLI2SQ_DIV_4

PLLI2S division factor for PLLI2SQ output by 4

LL_RCC_PLLI2SQ_DIV_5

PLLI2S division factor for PLLI2SQ output by 5

LL_RCC_PLLI2SQ_DIV_6

PLLI2S division factor for PLLI2SQ output by 6

LL_RCC_PLLI2SQ_DIV_7

PLLI2S division factor for PLLI2SQ output by 7

LL_RCC_PLLI2SQ_DIV_8

PLLI2S division factor for PLLI2SQ output by 8

LL_RCC_PLLI2SQ_DIV_9

PLLI2S division factor for PLLI2SQ output by 9

LL_RCC_PLLI2SQ_DIV_10

PLLI2S division factor for PLLI2SQ output by 10

LL_RCC_PLLI2SQ_DIV_11

PLLI2S division factor for PLLI2SQ output by 11

LL_RCC_PLLI2SQ_DIV_12

PLLI2S division factor for PLLI2SQ output by 12

LL_RCC_PLLI2SQ_DIV_13

PLLI2S division factor for PLLI2SQ output by 13

LL_RCC_PLLI2SQ_DIV_14

PLLI2S division factor for PLLI2SQ output by 14

LL_RCC_PLLI2SQ_DIV_15

PLLI2S division factor for PLLI2SQ output by 15

PLLI2SR division factor (PLLI2SR)

LL_RCC_PLLI2SR_DIV_2

PLLI2S division factor for PLLI2SR output by 2

LL_RCC_PLLI2SR_DIV_3

PLLI2S division factor for PLLI2SR output by 3

LL_RCC_PLLI2SR_DIV_4

PLLI2S division factor for PLLI2SR output by 4

LL_RCC_PLLI2SR_DIV_5

PLLI2S division factor for PLLI2SR output by 5

LL_RCC_PLLI2SR_DIV_6

PLLI2S division factor for PLLI2SR output by 6

LL_RCC_PLLI2SR_DIV_7

PLLI2S division factor for PLLI2SR output by 7

PLL, PLLI2S and PLLSAI division factor

LL_RCC_PLLM_DIV_2

PLL, PLLI2S and PLLSAI division factor by 2

LL_RCC_PLLM_DIV_3

PLL, PLLI2S and PLLSAI division factor by 3

LL_RCC_PLLM_DIV_4

PLL, PLLI2S and PLLSAI division factor by 4

LL_RCC_PLLM_DIV_5

PLL, PLLI2S and PLLSAI division factor by 5

LL_RCC_PLLM_DIV_6

PLL, PLLI2S and PLLSAI division factor by 6

LL_RCC_PLLM_DIV_7

PLL, PLLI2S and PLLSAI division factor by 7

LL_RCC_PLLM_DIV_8

PLL, PLLI2S and PLLSAI division factor by 8

LL_RCC_PLLM_DIV_9

PLL, PLLI2S and PLLSAI division factor by 9

LL_RCC_PLLM_DIV_10

PLL, PLLI2S and PLLSAI division factor by 10

LL_RCC_PLLM_DIV_11

PLL, PLLI2S and PLLSAI division factor by 11

LL_RCC_PLLM_DIV_12

PLL, PLLI2S and PLLSAI division factor by 12

LL_RCC_PLLM_DIV_13

PLL, PLLI2S and PLLSAI division factor by 13

LL_RCC_PLLM_DIV_14

PLL, PLLI2S and PLLSAI division factor by 14

LL_RCC_PLLM_DIV_15

PLL, PLLI2S and PLLSAI division factor by 15

LL_RCC_PLLM_DIV_16

PLL, PLLI2S and PLLSAI division factor by 16

LL_RCC_PLLM_DIV_17

PLL, PLLI2S and PLLSAI division factor by 17

LL_RCC_PLLM_DIV_18

PLL, PLLI2S and PLLSAI division factor by 18

LL_RCC_PLLM_DIV_19

PLL, PLLI2S and PLLSAI division factor by 19

LL_RCC_PLLM_DIV_20

PLL, PLLI2S and PLLSAI division factor by 20

LL_RCC_PLLM_DIV_21

PLL, PLLI2S and PLLSAI division factor by 21

LL_RCC_PLLM_DIV_22

PLL, PLLI2S and PLLSAI division factor by 22

LL_RCC_PLLM_DIV_23

PLL, PLLI2S and PLLSAI division factor by 23

LL_RCC_PLLM_DIV_24

PLL, PLLI2S and PLLSAI division factor by 24

LL_RCC_PLLM_DIV_25

PLL, PLLI2S and PLLSAI division factor by 25

LL_RCC_PLLM_DIV_26

PLL, PLLI2S and PLLSAI division factor by 26

LL_RCC_PLLM_DIV_27

PLL, PLLI2S and PLLSAI division factor by 27

LL_RCC_PLLM_DIV_28

PLL, PLLI2S and PLLSAI division factor by 28

LL_RCC_PLLM_DIV_29

PLL, PLLI2S and PLLSAI division factor by 29

LL_RCC_PLLM_DIV_30

PLL, PLLI2S and PLLSAI division factor by 30

LL_RCC_PLLM_DIV_31

PLL, PLLI2S and PLLSAI division factor by 31

LL_RCC_PLLM_DIV_32

PLL, PLLI2S and PLLSAI division factor by 32

LL_RCC_PLLM_DIV_33

PLL, PLLI2S and PLLSAI division factor by 33

LL_RCC_PLLM_DIV_34

PLL, PLLI2S and PLLSAI division factor by 34

LL_RCC_PLLM_DIV_35

PLL, PLLI2S and PLLSAI division factor by 35

LL_RCC_PLLM_DIV_36

PLL, PLLI2S and PLLSAI division factor by 36

LL_RCC_PLLM_DIV_37

PLL, PLLI2S and PLLSAI division factor by 37

LL_RCC_PLLM_DIV_38

PLL, PLLI2S and PLLSAI division factor by 38

LL_RCC_PLLM_DIV_39

PLL, PLLI2S and PLLSAI division factor by 39

LL_RCC_PLLM_DIV_40

PLL, PLLI2S and PLLSAI division factor by 40

LL_RCC_PLLM_DIV_41

PLL, PLLI2S and PLLSAI division factor by 41

LL_RCC_PLLM_DIV_42

PLL, PLLI2S and PLLSAI division factor by 42

LL_RCC_PLLM_DIV_43

PLL, PLLI2S and PLLSAI division factor by 43

LL_RCC_PLLM_DIV_44

PLL, PLLI2S and PLLSAI division factor by 44

LL_RCC_PLLM_DIV_45

PLL, PLLI2S and PLLSAI division factor by 45

LL_RCC_PLLM_DIV_46

PLL, PLLI2S and PLLSAI division factor by 46

LL_RCC_PLLM_DIV_47

PLL, PLLI2S and PLLSAI division factor by 47

LL_RCC_PLLM_DIV_48

PLL, PLLI2S and PLLSAI division factor by 48

LL_RCC_PLLM_DIV_49

PLL, PLLI2S and PLLSAI division factor by 49

LL_RCC_PLLM_DIV_50

PLL, PLLI2S and PLLSAI division factor by 50

LL_RCC_PLLM_DIV_51

PLL, PLLI2S and PLLSAI division factor by 51

LL_RCC_PLLM_DIV_52

PLL, PLLI2S and PLLSAI division factor by 52

LL_RCC_PLLM_DIV_53

PLL, PLLI2S and PLLSAI division factor by 53

LL_RCC_PLLM_DIV_54

PLL, PLLI2S and PLLSAI division factor by 54

LL_RCC_PLLM_DIV_55

PLL, PLLI2S and PLLSAI division factor by 55

LL_RCC_PLLM_DIV_56

PLL, PLLI2S and PLLSAI division factor by 56

LL_RCC_PLLM_DIV_57

PLL, PLLI2S and PLLSAI division factor by 57

LL_RCC_PLLM_DIV_58

PLL, PLLI2S and PLLSAI division factor by 58

LL_RCC_PLLM_DIV_59

PLL, PLLI2S and PLLSAI division factor by 59

LL_RCC_PLLM_DIV_60

PLL, PLLI2S and PLLSAI division factor by 60

LL_RCC_PLLM_DIV_61

PLL, PLLI2S and PLLSAI division factor by 61

LL_RCC_PLLM_DIV_62

PLL, PLLI2S and PLLSAI division factor by 62

LL_RCC_PLLM_DIV_63

PLL, PLLI2S and PLLSAI division factor by 63

PLL division factor (PLLP)

LL_RCC_PLLP_DIV_2

Main PLL division factor for PLLP output by 2

LL_RCC_PLLP_DIV_4

Main PLL division factor for PLLP output by 4

LL_RCC_PLLP_DIV_6

Main PLL division factor for PLLP output by 6

LL_RCC_PLLP_DIV_8

Main PLL division factor for PLLP output by 8

PLL division factor (PLLQ)

LL_RCC_PLLQ_DIV_2

Main PLL division factor for PLLQ output by 2

LL_RCC_PLLQ_DIV_3

Main PLL division factor for PLLQ output by 3

LL_RCC_PLLQ_DIV_4

Main PLL division factor for PLLQ output by 4

LL_RCC_PLLQ_DIV_5

Main PLL division factor for PLLQ output by 5

LL_RCC_PLLQ_DIV_6

Main PLL division factor for PLLQ output by 6

LL_RCC_PLLQ_DIV_7

Main PLL division factor for PLLQ output by 7

LL_RCC_PLLQ_DIV_8

Main PLL division factor for PLLQ output by 8

LL_RCC_PLLQ_DIV_9

Main PLL division factor for PLLQ output by 9

LL_RCC_PLLQ_DIV_10

Main PLL division factor for PLLQ output by 10

LL_RCC_PLLQ_DIV_11

Main PLL division factor for PLLQ output by 11

LL_RCC_PLLQ_DIV_12

Main PLL division factor for PLLQ output by 12

LL_RCC_PLLQ_DIV_13

Main PLL division factor for PLLQ output by 13

LL_RCC_PLLQ_DIV_14

Main PLL division factor for PLLQ output by 14

LL_RCC_PLLQ_DIV_15

Main PLL division factor for PLLQ output by 15

PLL division factor (PLLR)

LL_RCC_PLLR_DIV_2

Main PLL division factor for PLLCLK (system clock) by 2

LL_RCC_PLLR_DIV_3

Main PLL division factor for PLLCLK (system clock) by 3

LL_RCC_PLLR_DIV_4

Main PLL division factor for PLLCLK (system clock) by 4

LL_RCC_PLLR_DIV_5

Main PLL division factor for PLLCLK (system clock) by 5

LL_RCC_PLLR_DIV_6

Main PLL division factor for PLLCLK (system clock) by 6

LL_RCC_PLLR_DIV_7

Main PLL division factor for PLLCLK (system clock) by 7

PLLSAIDIVQ division factor (PLLSAIDIVQ)

LL_RCC_PLLSAIDIVQ_DIV_1

PLLSAI division factor for PLLSAIDIVQ output by 1

LL_RCC_PLLSAIDIVQ_DIV_2

PLLSAI division factor for PLLSAIDIVQ output by 2

LL_RCC_PLLSAIDIVQ_DIV_3

PLLSAI division factor for PLLSAIDIVQ output by 3

LL_RCC_PLLSAIDIVQ_DIV_4

PLLSAI division factor for PLLSAIDIVQ output by 4

LL_RCC_PLLSAIDIVQ_DIV_5

PLLSAI division factor for PLLSAIDIVQ output by 5

LL_RCC_PLLSAIDIVQ_DIV_6

PLLSAI division factor for PLLSAIDIVQ output by 6

LL_RCC_PLLSAIDIVQ_DIV_7

PLLSAI division factor for PLLSAIDIVQ output by 7

LL_RCC_PLLSAIDIVQ_DIV_8

PLLSAI division factor for PLLSAIDIVQ output by 8

LL_RCC_PLLSAIDIVQ_DIV_9

PLLSAI division factor for PLLSAIDIVQ output by 9

LL_RCC_PLLSAIDIVQ_DIV_10

PLLSAI division factor for PLLSAIDIVQ output by 10

LL_RCC_PLLSAIDIVQ_DIV_11

PLLSAI division factor for PLLSAIDIVQ output by 11

LL_RCC_PLLSAIDIVQ_DIV_12

PLLSAI division factor for PLLSAIDIVQ output by 12

LL_RCC_PLLSAIDIVQ_DIV_13

PLLSAI division factor for PLLSAIDIVQ output by 13

LL_RCC_PLLSAIDIVQ_DIV_14

PLLSAI division factor for PLLSAIDIVQ output by 14

LL_RCC_PLLSAIDIVQ_DIV_15

PLLSAI division factor for PLLSAIDIVQ output by 15

LL_RCC_PLLSAIDIVQ_DIV_16

PLLSAI division factor for PLLSAIDIVQ output by 16

LL_RCC_PLLSAIDIVQ_DIV_17

PLLSAI division factor for PLLSAIDIVQ output by 17

LL_RCC_PLLSAIDIVQ_DIV_18

PLLSAI division factor for PLLSAIDIVQ output by 18

LL_RCC_PLLSAIDIVQ_DIV_19

PLLSAI division factor for PLLSAIDIVQ output by 19

LL_RCC_PLLSAIDIVQ_DIV_20

PLLSAI division factor for PLLSAIDIVQ output by 20

LL_RCC_PLLSAIDIVQ_DIV_21

PLLSAI division factor for PLLSAIDIVQ output by 21

LL_RCC_PLLSAIDIVQ_DIV_22

PLLSAI division factor for PLLSAIDIVQ output by 22

LL_RCC_PLLSAIDIVQ_DIV_23

PLLSAI division factor for PLLSAIDIVQ output by 23

LL_RCC_PLLSAIDIVQ_DIV_24

PLLSAI division factor for PLLSAIDIVQ output by 24

LL_RCC_PLLSAIDIVQ_DIV_25

PLLSAI division factor for PLLSAIDIVQ output by 25

LL_RCC_PLLSAIDIVQ_DIV_26

PLLSAI division factor for PLLSAIDIVQ output by 26

LL_RCC_PLLSAIDIVQ_DIV_27

PLLSAI division factor for PLLSAIDIVQ output by 27

LL_RCC_PLLSAIDIVQ_DIV_28

PLLSAI division factor for PLLSAIDIVQ output by 28

LL_RCC_PLLSAIDIVQ_DIV_29

PLLSAI division factor for PLLSAIDIVQ output by 29

LL_RCC_PLLSAIDIVQ_DIV_30

PLLSAI division factor for PLLSAIDIVQ output by 30

LL_RCC_PLLSAIDIVQ_DIV_31

PLLSAI division factor for PLLSAIDIVQ output by 31

LL_RCC_PLLSAIDIVQ_DIV_32

PLLSAI division factor for PLLSAIDIVQ output by 32

PLLSAIDIVR division factor (PLLSAIDIVR)

LL_RCC_PLLSAIDIVR_DIV_2

PLLSAI division factor for PLLSAIDIVR output by 2

LL_RCC_PLLSAIDIVR_DIV_4

PLLSAI division factor for PLLSAIDIVR output by 4

LL_RCC_PLLSAIDIVR_DIV_8

PLLSAI division factor for PLLSAIDIVR output by 8

LL_RCC_PLLSAIDIVR_DIV_16

PLLSAI division factor for PLLSAIDIVR output by 16

PLLSAIM division factor (PLLSAIM or PLLM)

LL_RCC_PLLSAIM_DIV_2

PLLSAI division factor for PLLSAIM output by 2

LL_RCC_PLLSAIM_DIV_3

PLLSAI division factor for PLLSAIM output by 3

LL_RCC_PLLSAIM_DIV_4

PLLSAI division factor for PLLSAIM output by 4

LL_RCC_PLLSAIM_DIV_5

PLLSAI division factor for PLLSAIM output by 5

LL_RCC_PLLSAIM_DIV_6

PLLSAI division factor for PLLSAIM output by 6

LL_RCC_PLLSAIM_DIV_7

PLLSAI division factor for PLLSAIM output by 7

LL_RCC_PLLSAIM_DIV_8

PLLSAI division factor for PLLSAIM output by 8

LL_RCC_PLLSAIM_DIV_9

PLLSAI division factor for PLLSAIM output by 9

LL_RCC_PLLSAIM_DIV_10

PLLSAI division factor for PLLSAIM output by 10

LL_RCC_PLLSAIM_DIV_11

PLLSAI division factor for PLLSAIM output by 11

LL_RCC_PLLSAIM_DIV_12

PLLSAI division factor for PLLSAIM output by 12

LL_RCC_PLLSAIM_DIV_13

PLLSAI division factor for PLLSAIM output by 13

LL_RCC_PLLSAIM_DIV_14

PLLSAI division factor for PLLSAIM output by 14

LL_RCC_PLLSAIM_DIV_15

PLLSAI division factor for PLLSAIM output by 15

LL_RCC_PLLSAIM_DIV_16

PLLSAI division factor for PLLSAIM output by 16

LL_RCC_PLLSAIM_DIV_17

PLLSAI division factor for PLLSAIM output by 17

LL_RCC_PLLSAIM_DIV_18

PLLSAI division factor for PLLSAIM output by 18

LL_RCC_PLLSAIM_DIV_19

PLLSAI division factor for PLLSAIM output by 19

LL_RCC_PLLSAIM_DIV_20

PLLSAI division factor for PLLSAIM output by 20

LL_RCC_PLLSAIM_DIV_21

PLLSAI division factor for PLLSAIM output by 21

LL_RCC_PLLSAIM_DIV_22

PLLSAI division factor for PLLSAIM output by 22

LL_RCC_PLLSAIM_DIV_23

PLLSAI division factor for PLLSAIM output by 23

LL_RCC_PLLSAIM_DIV_24

PLLSAI division factor for PLLSAIM output by 24

LL_RCC_PLLSAIM_DIV_25

PLLSAI division factor for PLLSAIM output by 25

LL_RCC_PLLSAIM_DIV_26

PLLSAI division factor for PLLSAIM output by 26

LL_RCC_PLLSAIM_DIV_27

PLLSAI division factor for PLLSAIM output by 27

LL_RCC_PLLSAIM_DIV_28

PLLSAI division factor for PLLSAIM output by 28

LL_RCC_PLLSAIM_DIV_29

PLLSAI division factor for PLLSAIM output by 29

LL_RCC_PLLSAIM_DIV_30

PLLSAI division factor for PLLSAIM output by 30

LL_RCC_PLLSAIM_DIV_31

PLLSAI division factor for PLLSAIM output by 31

LL_RCC_PLLSAIM_DIV_32

PLLSAI division factor for PLLSAIM output by 32

LL_RCC_PLLSAIM_DIV_33

PLLSAI division factor for PLLSAIM output by 33

LL_RCC_PLLSAIM_DIV_34

PLLSAI division factor for PLLSAIM output by 34

LL_RCC_PLLSAIM_DIV_35

PLLSAI division factor for PLLSAIM output by 35

LL_RCC_PLLSAIM_DIV_36

PLLSAI division factor for PLLSAIM output by 36

LL_RCC_PLLSAIM_DIV_37

PLLSAI division factor for PLLSAIM output by 37

LL_RCC_PLLSAIM_DIV_38

PLLSAI division factor for PLLSAIM output by 38

LL_RCC_PLLSAIM_DIV_39

PLLSAI division factor for PLLSAIM output by 39

LL_RCC_PLLSAIM_DIV_40

PLLSAI division factor for PLLSAIM output by 40

LL_RCC_PLLSAIM_DIV_41

PLLSAI division factor for PLLSAIM output by 41

LL_RCC_PLLSAIM_DIV_42

PLLSAI division factor for PLLSAIM output by 42

LL_RCC_PLLSAIM_DIV_43

PLLSAI division factor for PLLSAIM output by 43

LL_RCC_PLLSAIM_DIV_44

PLLSAI division factor for PLLSAIM output by 44

LL_RCC_PLLSAIM_DIV_45

PLLSAI division factor for PLLSAIM output by 45

LL_RCC_PLLSAIM_DIV_46

PLLSAI division factor for PLLSAIM output by 46

LL_RCC_PLLSAIM_DIV_47

PLLSAI division factor for PLLSAIM output by 47

LL_RCC_PLLSAIM_DIV_48

PLLSAI division factor for PLLSAIM output by 48

LL_RCC_PLLSAIM_DIV_49

PLLSAI division factor for PLLSAIM output by 49

LL_RCC_PLLSAIM_DIV_50

PLLSAI division factor for PLLSAIM output by 50

LL_RCC_PLLSAIM_DIV_51

PLLSAI division factor for PLLSAIM output by 51

LL_RCC_PLLSAIM_DIV_52

PLLSAI division factor for PLLSAIM output by 52

LL_RCC_PLLSAIM_DIV_53

PLLSAI division factor for PLLSAIM output by 53

LL_RCC_PLLSAIM_DIV_54

PLLSAI division factor for PLLSAIM output by 54

LL_RCC_PLLSAIM_DIV_55

PLLSAI division factor for PLLSAIM output by 55

LL_RCC_PLLSAIM_DIV_56

PLLSAI division factor for PLLSAIM output by 56

LL_RCC_PLLSAIM_DIV_57

PLLSAI division factor for PLLSAIM output by 57

LL_RCC_PLLSAIM_DIV_58

PLLSAI division factor for PLLSAIM output by 58

LL_RCC_PLLSAIM_DIV_59

PLLSAI division factor for PLLSAIM output by 59

LL_RCC_PLLSAIM_DIV_60

PLLSAI division factor for PLLSAIM output by 60

LL_RCC_PLLSAIM_DIV_61

PLLSAI division factor for PLLSAIM output by 61

LL_RCC_PLLSAIM_DIV_62

PLLSAI division factor for PLLSAIM output by 62

LL_RCC_PLLSAIM_DIV_63

PLLSAI division factor for PLLSAIM output by 63

PLLSAIP division factor (PLLSAIP)

LL_RCC_PLLSAIP_DIV_2

PLLSAI division factor for PLLSAIP output by 2

LL_RCC_PLLSAIP_DIV_4

PLLSAI division factor for PLLSAIP output by 4

LL_RCC_PLLSAIP_DIV_6

PLLSAI division factor for PLLSAIP output by 6

LL_RCC_PLLSAIP_DIV_8

PLLSAI division factor for PLLSAIP output by 8

PLLSAIQ division factor (PLLSAIQ)

LL_RCC_PLLSAIQ_DIV_2

PLLSAI division factor for PLLSAIQ output by 2

LL_RCC_PLLSAIQ_DIV_3

PLLSAI division factor for PLLSAIQ output by 3

LL_RCC_PLLSAIQ_DIV_4

PLLSAI division factor for PLLSAIQ output by 4

LL_RCC_PLLSAIQ_DIV_5

PLLSAI division factor for PLLSAIQ output by 5

LL_RCC_PLLSAIQ_DIV_6

PLLSAI division factor for PLLSAIQ output by 6

LL_RCC_PLLSAIQ_DIV_7

PLLSAI division factor for PLLSAIQ output by 7

LL_RCC_PLLSAIQ_DIV_8

PLLSAI division factor for PLLSAIQ output by 8

LL_RCC_PLLSAIQ_DIV_9

PLLSAI division factor for PLLSAIQ output by 9

LL_RCC_PLLSAIQ_DIV_10

PLLSAI division factor for PLLSAIQ output by 10

LL_RCC_PLLSAIQ_DIV_11

PLLSAI division factor for PLLSAIQ output by 11

LL_RCC_PLLSAIQ_DIV_12

PLLSAI division factor for PLLSAIQ output by 12

LL_RCC_PLLSAIQ_DIV_13

PLLSAI division factor for PLLSAIQ output by 13

LL_RCC_PLLSAIQ_DIV_14

PLLSAI division factor for PLLSAIQ output by 14

LL_RCC_PLLSAIQ_DIV_15

PLLSAI division factor for PLLSAIQ output by 15

PLLSAIR division factor (PLLSAIR)

LL_RCC_PLLSAIR_DIV_2

PLLSAI division factor for PLLSAIR output by 2

LL_RCC_PLLSAIR_DIV_3

PLLSAI division factor for PLLSAIR output by 3

LL_RCC_PLLSAIR_DIV_4

PLLSAI division factor for PLLSAIR output by 4

LL_RCC_PLLSAIR_DIV_5

PLLSAI division factor for PLLSAIR output by 5

LL_RCC_PLLSAIR_DIV_6

PLLSAI division factor for PLLSAIR output by 6

LL_RCC_PLLSAIR_DIV_7

PLLSAI division factor for PLLSAIR output by 7

PLL, PLLI2S and PLLSAI entry clock source

LL_RCC_PLLSOURCE_HSI

HSI16 clock selected as PLL entry clock source

LL_RCC_PLLSOURCE_HSE

HSE clock selected as PLL entry clock source

PLL Spread Spectrum Selection

LL_RCC_SPREAD_SELECT_CENTER

PLL center spread spectrum selection

LL_RCC_SPREAD_SELECT_DOWN

PLL down spread spectrum selection

Peripheral RNG get clock source

LL_RCC_RNG_CLKSOURCE

RNG Clock source selection

Peripheral RNG clock source selection

LL_RCC_RNG_CLKSOURCE_PLL

PLL clock used as RNG clock source

LL_RCC_RNG_CLKSOURCE_PLLSAI

PLLSAI clock used as RNG clock source

RTC clock source selection

LL_RCC_RTC_CLKSOURCE_NONE

No clock used as RTC clock

LL_RCC_RTC_CLKSOURCE_LSE

LSE oscillator clock used as RTC clock

LL_RCC_RTC_CLKSOURCE_LSI

LSI oscillator clock used as RTC clock

LL_RCC_RTC_CLKSOURCE_HSE

HSE oscillator clock divided by HSE prescaler used as RTC clock

HSE prescaler for RTC clock

LL_RCC_RTC_NOCLOCK

HSE not divided

LL_RCC_RTC_HSE_DIV_2

HSE clock divided by 2

LL_RCC_RTC_HSE_DIV_3

HSE clock divided by 3

LL_RCC_RTC_HSE_DIV_4

HSE clock divided by 4

LL_RCC_RTC_HSE_DIV_5

HSE clock divided by 5

LL_RCC_RTC_HSE_DIV_6

HSE clock divided by 6

LL_RCC_RTC_HSE_DIV_7

HSE clock divided by 7

LL_RCC_RTC_HSE_DIV_8

HSE clock divided by 8

LL_RCC_RTC_HSE_DIV_9

HSE clock divided by 9

LL_RCC_RTC_HSE_DIV_10

HSE clock divided by 10

LL_RCC_RTC_HSE_DIV_11

HSE clock divided by 11

LL_RCC_RTC_HSE_DIV_12

HSE clock divided by 12

LL_RCC_RTC_HSE_DIV_13

HSE clock divided by 13

LL_RCC_RTC_HSE_DIV_14

HSE clock divided by 14

LL_RCC_RTC_HSE_DIV_15

HSE clock divided by 15

LL_RCC_RTC_HSE_DIV_16

HSE clock divided by 16

LL_RCC_RTC_HSE_DIV_17

HSE clock divided by 17

LL_RCC_RTC_HSE_DIV_18

HSE clock divided by 18

LL_RCC_RTC_HSE_DIV_19

HSE clock divided by 19

LL_RCC_RTC_HSE_DIV_20

HSE clock divided by 20

LL_RCC_RTC_HSE_DIV_21

HSE clock divided by 21

LL_RCC_RTC_HSE_DIV_22

HSE clock divided by 22

LL_RCC_RTC_HSE_DIV_23

HSE clock divided by 23

LL_RCC_RTC_HSE_DIV_24

HSE clock divided by 24

LL_RCC_RTC_HSE_DIV_25

HSE clock divided by 25

LL_RCC_RTC_HSE_DIV_26

HSE clock divided by 26

LL_RCC_RTC_HSE_DIV_27

HSE clock divided by 27

LL_RCC_RTC_HSE_DIV_28

HSE clock divided by 28

LL_RCC_RTC_HSE_DIV_29

HSE clock divided by 29

LL_RCC_RTC_HSE_DIV_30

HSE clock divided by 30

LL_RCC_RTC_HSE_DIV_31

HSE clock divided by 31

Peripheral SAI get clock source

LL_RCC_SAI1_A_CLKSOURCE

SAI1 block A Clock source selection

LL_RCC_SAI1_B_CLKSOURCE

SAI1 block B Clock source selection

Peripheral SAI clock source selection

LL_RCC_SAI1_A_CLKSOURCE_PLLSAI

PLLSAI clock used as SAI1 block A clock source

LL_RCC_SAI1_A_CLKSOURCE_PLLI2S

PLLI2S clock used as SAI1 block A clock source

LL_RCC_SAI1_A_CLKSOURCE_PIN

External pin clock used as SAI1 block A clock source

LL_RCC_SAI1_B_CLKSOURCE_PLLSAI

PLLSAI clock used as SAI1 block B clock source

LL_RCC_SAI1_B_CLKSOURCE_PLLI2S

PLLI2S clock used as SAI1 block B clock source

LL_RCC_SAI1_B_CLKSOURCE_PIN

External pin clock used as SAI1 block B clock source

Peripheral SDIO get clock source

LL_RCC_SDIO_CLKSOURCE

SDIO Clock source selection

Peripheral SDIO clock source selection

LL_RCC_SDIO_CLKSOURCE_PLL48CLK

PLL 48M domain clock used as SDIO clock

LL_RCC_SDIO_CLKSOURCE_SYSCLK

System clock clock used as SDIO clock

AHB prescaler

LL_RCC_SYSCLK_DIV_1

SYSCLK not divided

LL_RCC_SYSCLK_DIV_2

SYSCLK divided by 2

LL_RCC_SYSCLK_DIV_4

SYSCLK divided by 4

LL_RCC_SYSCLK_DIV_8

SYSCLK divided by 8

LL_RCC_SYSCLK_DIV_16

SYSCLK divided by 16

LL_RCC_SYSCLK_DIV_64

SYSCLK divided by 64

LL_RCC_SYSCLK_DIV_128

SYSCLK divided by 128

LL_RCC_SYSCLK_DIV_256

SYSCLK divided by 256

LL_RCC_SYSCLK_DIV_512

SYSCLK divided by 512

System clock switch

LL_RCC_SYS_CLKSOURCE_HSI

HSI selection as system clock

LL_RCC_SYS_CLKSOURCE_HSE

HSE selection as system clock

LL_RCC_SYS_CLKSOURCE_PLL

PLL selection as system clock

System clock switch status

LL_RCC_SYS_CLKSOURCE_STATUS_HSI

HSI used as system clock

LL_RCC_SYS_CLKSOURCE_STATUS_HSE

HSE used as system clock

LL_RCC_SYS_CLKSOURCE_STATUS_PLL

PLL used as system clock

Timers clocks prescalers selection

LL_RCC_TIM_PRESCALER_TWICE

Timers clock to twice PCLK

LL_RCC_TIM_PRESCALER_FOUR_TIMES

Timers clock to four time PCLK

Peripheral USB get clock source

LL_RCC_USB_CLKSOURCE

USB Clock source selection

Peripheral USB clock source selection

LL_RCC_USB_CLKSOURCE_PLL

PLL clock used as USB clock source

LL_RCC_USB_CLKSOURCE_PLLSAI

PLLSAI clock used as USB clock source

Calculate frequencies

__LL_RCC_CALC_PLLCLK_FREQ

Description:

- Helper macro to calculate the PLLCLK frequency on system domain.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51

Return value:

- PLL: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLCLK_FREQ (HSE_VALUE,LL_RCC_PLL_GetDivider (), LL_RCC_PLL_GetN (), LL_RCC_PLL_GetP ());

__LL_RCC_CALC_PLLCLK_48M_FREQ

Description:

- Helper macro to calculate the PLLCLK frequency used on 48M domain.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51

Return value:

- PLL: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLCLK_48M_FREQ (HSE_VALUE,LL_RCC_PLL_GetDivider (), LL_RCC_PLL_GetN (), LL_RCC_PLL_GetQ ());

__LL_RCC_CALC_PLLCLK_DSI_FREQ

Description:

- Helper macro to calculate the PLLCLK frequency used on DSI.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51

Return value:

- PLL: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLCLK_DSI_FREQ (HSE_VALUE, LL_RCC_PLL_GetDivider (), LL_RCC_PLL_GetN (), LL_RCC_PLL_GetR ());

__LL_RCC_CALC_PLLCLK_SAI_FREQ

Description:

- Helper macro to calculate the PLLCLK frequency used on SAI.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLM_DIV_2
  - LL_RCC_PLLM_DIV_3
  - LL_RCC_PLLM_DIV_4
  - LL_RCC_PLLM_DIV_5
  - LL_RCC_PLLM_DIV_6
  - LL_RCC_PLLM_DIV_7
  - LL_RCC_PLLM_DIV_8
  - LL_RCC_PLLM_DIV_9
  - LL_RCC_PLLM_DIV_10
  - LL_RCC_PLLM_DIV_11
  - LL_RCC_PLLM_DIV_12
  - LL_RCC_PLLM_DIV_13
  - LL_RCC_PLLM_DIV_14
  - LL_RCC_PLLM_DIV_15
  - LL_RCC_PLLM_DIV_16
  - LL_RCC_PLLM_DIV_17
  - LL_RCC_PLLM_DIV_18
  - LL_RCC_PLLM_DIV_19
  - LL_RCC_PLLM_DIV_20
  - LL_RCC_PLLM_DIV_21
  - LL_RCC_PLLM_DIV_22
  - LL_RCC_PLLM_DIV_23
  - LL_RCC_PLLM_DIV_24
  - LL_RCC_PLLM_DIV_25
  - LL_RCC_PLLM_DIV_26
  - LL_RCC_PLLM_DIV_27
  - LL_RCC_PLLM_DIV_28
  - LL_RCC_PLLM_DIV_29
  - LL_RCC_PLLM_DIV_30
  - LL_RCC_PLLM_DIV_31
  - LL_RCC_PLLM_DIV_32
  - LL_RCC_PLLM_DIV_33
  - LL_RCC_PLLM_DIV_34
  - LL_RCC_PLLM_DIV_35
  - LL_RCC_PLLM_DIV_36
  - LL_RCC_PLLM_DIV_37
  - LL_RCC_PLLM_DIV_38
  - LL_RCC_PLLM_DIV_39
  - LL_RCC_PLLM_DIV_40
  - LL_RCC_PLLM_DIV_41
  - LL_RCC_PLLM_DIV_42
  - LL_RCC_PLLM_DIV_43
  - LL_RCC_PLLM_DIV_44
  - LL_RCC_PLLM_DIV_45
  - LL_RCC_PLLM_DIV_46
  - LL_RCC_PLLM_DIV_47
  - LL_RCC_PLLM_DIV_48
  - LL_RCC_PLLM_DIV_49
  - LL_RCC_PLLM_DIV_50
  - LL_RCC_PLLM_DIV_51

Return value:

- PLL: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLCLK_SAI_FREQ (HSE_VALUE, LL_RCC_PLL_GetDivider (), LL_RCC_PLL_GetN (), LL_RCC_PLL_GetR (), LL_RCC_PLL_GetDIVR ());

__LL_RCC_CALC_PLLSAI_SAI_FREQ

Description:

- Helper macro to calculate the PLLSAI frequency used for SAI domain.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLSAIM_DIV_2
  - LL_RCC_PLLSAIM_DIV_3
  - LL_RCC_PLLSAIM_DIV_4
  - LL_RCC_PLLSAIM_DIV_5
  - LL_RCC_PLLSAIM_DIV_6
  - LL_RCC_PLLSAIM_DIV_7
  - LL_RCC_PLLSAIM_DIV_8
  - LL_RCC_PLLSAIM_DIV_9
  - LL_RCC_PLLSAIM_DIV_10
  - LL_RCC_PLLSAIM_DIV_11
  - LL_RCC_PLLSAIM_DIV_12
  - LL_RCC_PLLSAIM_DIV_13
  - LL_RCC_PLLSAIM_DIV_14
  - LL_RCC_PLLSAIM_DIV_15
  - LL_RCC_PLLSAIM_DIV_16
  - LL_RCC_PLLSAIM_DIV_17
  - LL_RCC_PLLSAIM_DIV_18
  - LL_RCC_PLLSAIM_DIV_19
  - LL_RCC_PLLSAIM_DIV_20
  - LL_RCC_PLLSAIM_DIV_21
  - LL_RCC_PLLSAIM_DIV_22
  - LL_RCC_PLLSAIM_DIV_23
  - LL_RCC_PLLSAIM_DIV_24
  - LL_RCC_PLLSAIM_DIV_25
  - LL_RCC_PLLSAIM_DIV_26
  - LL_RCC_PLLSAIM_DIV_27
  - LL_RCC_PLLSAIM_DIV_28
  - LL_RCC_PLLSAIM_DIV_29
  - LL_RCC_PLLSAIM_DIV_30
  - LL_RCC_PLLSAIM_DIV_31
  - LL_RCC_PLLSAIM_DIV_32
  - LL_RCC_PLLSAIM_DIV_33
  - LL_RCC_PLLSAIM_DIV_34
  - LL_RCC_PLLSAIM_DIV_35
  - LL_RCC_PLLSAIM_DIV_36
  - LL_RCC_PLLSAIM_DIV_37
  - LL_RCC_PLLSAIM_DIV_38
  - LL_RCC_PLLSAIM_DIV_39
  - LL_RCC_PLLSAIM_DIV_40
  - LL_RCC_PLLSAIM_DIV_41
  - LL_RCC_PLLSAIM_DIV_42
  - LL_RCC_PLLSAIM_DIV_43
  - LL_RCC_PLLSAIM_DIV_44
  - LL_RCC_PLLSAIM_DIV_45
  - LL_RCC_PLLSAIM_DIV_46
  - LL_RCC_PLLSAIM_DIV_47
  - LL_RCC_PLLSAIM_DIV_48
  - LL_RCC_PLLSAIM_DIV_49
  - LL_RCC_PLLSAIM_DIV_50
  - LL_RCC_PLLSAIM_DIV_51

Return value:

- PLLSAI: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLSAI_SAI_FREQ (HSE_VALUE,LL_RCC_PLLSAI_GetDivider (), LL_RCC_PLLSAI_GetN (), LL_RCC_PLLSAI_GetQ (), LL_RCC_PLLSAI_GetDIVQ ());

__LL_RCC_CALC_PLLSAI_48M_FREQ

Description:

- Helper macro to calculate the PLLSAI frequency used on 48Mhz domain.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLSAIM_DIV_2
  - LL_RCC_PLLSAIM_DIV_3
  - LL_RCC_PLLSAIM_DIV_4
  - LL_RCC_PLLSAIM_DIV_5
  - LL_RCC_PLLSAIM_DIV_6
  - LL_RCC_PLLSAIM_DIV_7
  - LL_RCC_PLLSAIM_DIV_8
  - LL_RCC_PLLSAIM_DIV_9
  - LL_RCC_PLLSAIM_DIV_10
  - LL_RCC_PLLSAIM_DIV_11
  - LL_RCC_PLLSAIM_DIV_12
  - LL_RCC_PLLSAIM_DIV_13
  - LL_RCC_PLLSAIM_DIV_14
  - LL_RCC_PLLSAIM_DIV_15
  - LL_RCC_PLLSAIM_DIV_16
  - LL_RCC_PLLSAIM_DIV_17
  - LL_RCC_PLLSAIM_DIV_18
  - LL_RCC_PLLSAIM_DIV_19
  - LL_RCC_PLLSAIM_DIV_20
  - LL_RCC_PLLSAIM_DIV_21
  - LL_RCC_PLLSAIM_DIV_22
  - LL_RCC_PLLSAIM_DIV_23
  - LL_RCC_PLLSAIM_DIV_24
  - LL_RCC_PLLSAIM_DIV_25
  - LL_RCC_PLLSAIM_DIV_26
  - LL_RCC_PLLSAIM_DIV_27
  - LL_RCC_PLLSAIM_DIV_28
  - LL_RCC_PLLSAIM_DIV_29
  - LL_RCC_PLLSAIM_DIV_30
  - LL_RCC_PLLSAIM_DIV_31
  - LL_RCC_PLLSAIM_DIV_32
  - LL_RCC_PLLSAIM_DIV_33
  - LL_RCC_PLLSAIM_DIV_34
  - LL_RCC_PLLSAIM_DIV_35
  - LL_RCC_PLLSAIM_DIV_36
  - LL_RCC_PLLSAIM_DIV_37
  - LL_RCC_PLLSAIM_DIV_38
  - LL_RCC_PLLSAIM_DIV_39
  - LL_RCC_PLLSAIM_DIV_40
  - LL_RCC_PLLSAIM_DIV_41
  - LL_RCC_PLLSAIM_DIV_42
  - LL_RCC_PLLSAIM_DIV_43
  - LL_RCC_PLLSAIM_DIV_44
  - LL_RCC_PLLSAIM_DIV_45
  - LL_RCC_PLLSAIM_DIV_46
  - LL_RCC_PLLSAIM_DIV_47
  - LL_RCC_PLLSAIM_DIV_48
  - LL_RCC_PLLSAIM_DIV_49
  - LL_RCC_PLLSAIM_DIV_50
  - LL_RCC_PLLSAIM_DIV_51

Return value:

- PLLSAI: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLSAI_48M_FREQ (HSE_VALUE,LL_RCC_PLLSAI_GetDivider (), LL_RCC_PLLSAI_GetN (), LL_RCC_PLLSAI_GetP ());

__LL_RCC_CALC_PLLSAI_LTDC_FREQ

Description:

- Helper macro to calculate the PLLSAI frequency used for LTDC domain.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLSAIM_DIV_2
  - LL_RCC_PLLSAIM_DIV_3
  - LL_RCC_PLLSAIM_DIV_4
  - LL_RCC_PLLSAIM_DIV_5
  - LL_RCC_PLLSAIM_DIV_6
  - LL_RCC_PLLSAIM_DIV_7
  - LL_RCC_PLLSAIM_DIV_8
  - LL_RCC_PLLSAIM_DIV_9
  - LL_RCC_PLLSAIM_DIV_10
  - LL_RCC_PLLSAIM_DIV_11
  - LL_RCC_PLLSAIM_DIV_12
  - LL_RCC_PLLSAIM_DIV_13
  - LL_RCC_PLLSAIM_DIV_14
  - LL_RCC_PLLSAIM_DIV_15
  - LL_RCC_PLLSAIM_DIV_16
  - LL_RCC_PLLSAIM_DIV_17
  - LL_RCC_PLLSAIM_DIV_18
  - LL_RCC_PLLSAIM_DIV_19
  - LL_RCC_PLLSAIM_DIV_20
  - LL_RCC_PLLSAIM_DIV_21
  - LL_RCC_PLLSAIM_DIV_22
  - LL_RCC_PLLSAIM_DIV_23
  - LL_RCC_PLLSAIM_DIV_24
  - LL_RCC_PLLSAIM_DIV_25
  - LL_RCC_PLLSAIM_DIV_26
  - LL_RCC_PLLSAIM_DIV_27
  - LL_RCC_PLLSAIM_DIV_28
  - LL_RCC_PLLSAIM_DIV_29
  - LL_RCC_PLLSAIM_DIV_30
  - LL_RCC_PLLSAIM_DIV_31
  - LL_RCC_PLLSAIM_DIV_32
  - LL_RCC_PLLSAIM_DIV_33
  - LL_RCC_PLLSAIM_DIV_34
  - LL_RCC_PLLSAIM_DIV_35
  - LL_RCC_PLLSAIM_DIV_36
  - LL_RCC_PLLSAIM_DIV_37
  - LL_RCC_PLLSAIM_DIV_38
  - LL_RCC_PLLSAIM_DIV_39
  - LL_RCC_PLLSAIM_DIV_40
  - LL_RCC_PLLSAIM_DIV_41
  - LL_RCC_PLLSAIM_DIV_42
  - LL_RCC_PLLSAIM_DIV_43
  - LL_RCC_PLLSAIM_DIV_44
  - LL_RCC_PLLSAIM_DIV_45
  - LL_RCC_PLLSAIM_DIV_46
  - LL_RCC_PLLSAIM_DIV_47
  - LL_RCC_PLLSAIM_DIV_48
  - LL_RCC_PLLSAIM_DIV_49
  - LL_RCC_PLLSAIM_DIV_50
  - LL_RCC_PLLSAIM_DIV_51

Return value:

- PLLSAI: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLSAI_LTDC_FREQ (HSE_VALUE,LL_RCC_PLLSAI_GetDivider (), LL_RCC_PLLSAI_GetN (), LL_RCC_PLLSAI_GetR (), LL_RCC_PLLSAI_GetDIVR ());

__LL_RCC_CALC_PLLI2S_SAI_FREQ

Description:

- Helper macro to calculate the PLLI2S frequency used for SAI domain.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLI2SM_DIV_2
  - LL_RCC_PLLI2SM_DIV_3
  - LL_RCC_PLLI2SM_DIV_4
  - LL_RCC_PLLI2SM_DIV_5
  - LL_RCC_PLLI2SM_DIV_6
  - LL_RCC_PLLI2SM_DIV_7
  - LL_RCC_PLLI2SM_DIV_8
  - LL_RCC_PLLI2SM_DIV_9
  - LL_RCC_PLLI2SM_DIV_10
  - LL_RCC_PLLI2SM_DIV_11
  - LL_RCC_PLLI2SM_DIV_12
  - LL_RCC_PLLI2SM_DIV_13
  - LL_RCC_PLLI2SM_DIV_14
  - LL_RCC_PLLI2SM_DIV_15
  - LL_RCC_PLLI2SM_DIV_16
  - LL_RCC_PLLI2SM_DIV_17
  - LL_RCC_PLLI2SM_DIV_18
  - LL_RCC_PLLI2SM_DIV_19
  - LL_RCC_PLLI2SM_DIV_20
  - LL_RCC_PLLI2SM_DIV_21
  - LL_RCC_PLLI2SM_DIV_22
  - LL_RCC_PLLI2SM_DIV_23
  - LL_RCC_PLLI2SM_DIV_24
  - LL_RCC_PLLI2SM_DIV_25
  - LL_RCC_PLLI2SM_DIV_26
  - LL_RCC_PLLI2SM_DIV_27
  - LL_RCC_PLLI2SM_DIV_28
  - LL_RCC_PLLI2SM_DIV_29
  - LL_RCC_PLLI2SM_DIV_30
  - LL_RCC_PLLI2SM_DIV_31
  - LL_RCC_PLLI2SM_DIV_32
  - LL_RCC_PLLI2SM_DIV_33
  - LL_RCC_PLLI2SM_DIV_34
  - LL_RCC_PLLI2SM_DIV_35
  - LL_RCC_PLLI2SM_DIV_36
  - LL_RCC_PLLI2SM_DIV_37
  - LL_RCC_PLLI2SM_DIV_38
  - LL_RCC_PLLI2SM_DIV_39
  - LL_RCC_PLLI2SM_DIV_40
  - LL_RCC_PLLI2SM_DIV_41
  - LL_RCC_PLLI2SM_DIV_42
  - LL_RCC_PLLI2SM_DIV_43
  - LL_RCC_PLLI2SM_DIV_44
  - LL_RCC_PLLI2SM_DIV_45
  - LL_RCC_PLLI2SM_DIV_46
  - LL_RCC_PLLI2SM_DIV_47
  - LL_RCC_PLLI2SM_DIV_48
  - LL_RCC_PLLI2SM_DIV_49
  - LL_RCC_PLLI2SM_DIV_50
  - LL_RCC_PLLI2SM_DIV_51

Return value:

- PLLI2S: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLI2S_SAI_FREQ (HSE_VALUE,LL_RCC_PLLI2S_GetDivider (), LL_RCC_PLLI2S_GetN (), LL_RCC_PLLI2S_GetQ (), LL_RCC_PLLI2S_GetDIVQ ());

__LL_RCC_CALC_PLLI2S_I2S_FREQ

Description:

- Helper macro to calculate the PLLI2S frequency used for I2S domain.

Parameters:

- __INPUTFREQ__: PLL Input frequency (based on HSE/HSI)
- __PLLM__: This parameter can be one of the following values:
  - LL_RCC_PLLI2SM_DIV_2
  - LL_RCC_PLLI2SM_DIV_3
  - LL_RCC_PLLI2SM_DIV_4
  - LL_RCC_PLLI2SM_DIV_5
  - LL_RCC_PLLI2SM_DIV_6
  - LL_RCC_PLLI2SM_DIV_7
  - LL_RCC_PLLI2SM_DIV_8
  - LL_RCC_PLLI2SM_DIV_9
  - LL_RCC_PLLI2SM_DIV_10
  - LL_RCC_PLLI2SM_DIV_11
  - LL_RCC_PLLI2SM_DIV_12
  - LL_RCC_PLLI2SM_DIV_13
  - LL_RCC_PLLI2SM_DIV_14
  - LL_RCC_PLLI2SM_DIV_15
  - LL_RCC_PLLI2SM_DIV_16
  - LL_RCC_PLLI2SM_DIV_17
  - LL_RCC_PLLI2SM_DIV_18
  - LL_RCC_PLLI2SM_DIV_19
  - LL_RCC_PLLI2SM_DIV_20
  - LL_RCC_PLLI2SM_DIV_21
  - LL_RCC_PLLI2SM_DIV_22
  - LL_RCC_PLLI2SM_DIV_23
  - LL_RCC_PLLI2SM_DIV_24
  - LL_RCC_PLLI2SM_DIV_25
  - LL_RCC_PLLI2SM_DIV_26
  - LL_RCC_PLLI2SM_DIV_27
  - LL_RCC_PLLI2SM_DIV_28
  - LL_RCC_PLLI2SM_DIV_29
  - LL_RCC_PLLI2SM_DIV_30
  - LL_RCC_PLLI2SM_DIV_31
  - LL_RCC_PLLI2SM_DIV_32
  - LL_RCC_PLLI2SM_DIV_33
  - LL_RCC_PLLI2SM_DIV_34
  - LL_RCC_PLLI2SM_DIV_35
  - LL_RCC_PLLI2SM_DIV_36
  - LL_RCC_PLLI2SM_DIV_37
  - LL_RCC_PLLI2SM_DIV_38
  - LL_RCC_PLLI2SM_DIV_39
  - LL_RCC_PLLI2SM_DIV_40
  - LL_RCC_PLLI2SM_DIV_41
  - LL_RCC_PLLI2SM_DIV_42
  - LL_RCC_PLLI2SM_DIV_43
  - LL_RCC_PLLI2SM_DIV_44
  - LL_RCC_PLLI2SM_DIV_45
  - LL_RCC_PLLI2SM_DIV_46
  - LL_RCC_PLLI2SM_DIV_47
  - LL_RCC_PLLI2SM_DIV_48
  - LL_RCC_PLLI2SM_DIV_49
  - LL_RCC_PLLI2SM_DIV_50
  - LL_RCC_PLLI2SM_DIV_51

Return value:

- PLLI2S: clock frequency (in Hz)

Notes:

- ex: __LL_RCC_CALC_PLLI2S_I2S_FREQ (HSE_VALUE,LL_RCC_PLLI2S_GetDivider (), LL_RCC_PLLI2S_GetN (), LL_RCC_PLLI2S_GetR ());

__LL_RCC_CALC_HCLK_FREQ

Description:

- Helper macro to calculate the HCLK frequency.

Parameters:

- __SYSCLKFREQ__: SYSCLK frequency (based on HSE/HSI/PLLCLK)
- __AHBPRESCALER__: This parameter can be one of the following values:
  - LL_RCC_SYSCLK_DIV_1
  - LL_RCC_SYSCLK_DIV_2
  - LL_RCC_SYSCLK_DIV_4
  - LL_RCC_SYSCLK_DIV_8
  - LL_RCC_SYSCLK_DIV_16
  - LL_RCC_SYSCLK_DIV_64
  - LL_RCC_SYSCLK_DIV_128
  - LL_RCC_SYSCLK_DIV_256
  - LL_RCC_SYSCLK_DIV_512

Return value:

- HCLK: clock frequency (in Hz)

__LL_RCC_CALC_PCLK1_FREQ

Description:

- Helper macro to calculate the PCLK1 frequency (ABP1)

Parameters:

- __HCLKFREQ__: HCLK frequency
- __APB1PRESCALER__: This parameter can be one of the following values:
  - LL_RCC_APB1_DIV_1
  - LL_RCC_APB1_DIV_2
  - LL_RCC_APB1_DIV_4
  - LL_RCC_APB1_DIV_8
  - LL_RCC_APB1_DIV_16

Return value:

- PCLK1: clock frequency (in Hz)

__LL_RCC_CALC_PCLK2_FREQ

Description:

- Helper macro to calculate the PCLK2 frequency (ABP2)

Parameters:

- __HCLKFREQ__: HCLK frequency
- __APB2PRESCALER__: This parameter can be one of the following values:
  - LL_RCC_APB2_DIV_1
  - LL_RCC_APB2_DIV_2
  - LL_RCC_APB2_DIV_4
  - LL_RCC_APB2_DIV_8
  - LL_RCC_APB2_DIV_16

Return value:

- PCLK2: clock frequency (in Hz)

Common Write and read registers Macros

LL_RCC_WriteReg

Description:

- Write a value in RCC register.

Parameters:

- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_RCC_ReadReg

Description:

- Read a value in RCC register.

Parameters:

- __REG__: Register to be read

Return value:

- Register: value

88 LL RNG Generic Driver

88.1 RNG Firmware driver API description

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

88.1.1 Detailed description of functions

LL_RNG_Enable

Function name

__STATIC_INLINE void LL_RNG_Enable (RNG_TypeDef * RNGx)

Function description

Enable Random Number Generation.

Parameters

- **RNGx:** RNG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR RNGEN LL_RNG_Enable

LL_RNG_Disable

Function name

__STATIC_INLINE void LL_RNG_Disable (RNG_TypeDef * RNGx)

Function description

Disable Random Number Generation.

Parameters

- **RNGx:** RNG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR RNGEN LL_RNG_Disable

LL_RNG_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_RNG_IsEnabled (RNG_TypeDef * RNGx)

Function description

Check if Random Number Generator is enabled.

Parameters

- **RNGx:** RNG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR RNGEN LL_RNG_IsEnabled

LL_RNG_IsActiveFlag_DRDY

Function name

__STATIC_INLINE uint32_t LL_RNG_IsActiveFlag_DRDY (RNG_TypeDef * RNGx)

Function description

Indicate if the RNG Data ready Flag is set or not.

Parameters

- **RNGx:** RNG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR DRDY LL_RNG_IsActiveFlag_DRDY

LL_RNG_IsActiveFlag_CECS

Function name

__STATIC_INLINE uint32_t LL_RNG_IsActiveFlag_CECS (RNG_TypeDef * RNGx)

Function description

Indicate if the Clock Error Current Status Flag is set or not.

Parameters

- **RNGx:** RNG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CECS LL_RNG_IsActiveFlag_CECS

LL_RNG_IsActiveFlag_SECS

Function name

__STATIC_INLINE uint32_t LL_RNG_IsActiveFlag_SECS (RNG_TypeDef * RNGx)

Function description

Indicate if the Seed Error Current Status Flag is set or not.

Parameters

- **RNGx:** RNG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR SECS LL_RNG_IsActiveFlag_SECS

LL_RNG_IsActiveFlag_CEIS

Function name

__STATIC_INLINE uint32_t LL_RNG_IsActiveFlag_CEIS (RNG_TypeDef * RNGx)

Function description

Indicate if the Clock Error Interrupt Status Flag is set or not.

Parameters

- **RNGx:** RNG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CEIS LL_RNG_IsActiveFlag_CEIS

LL_RNG_IsActiveFlag_SEIS

Function name

__STATIC_INLINE uint32_t LL_RNG_IsActiveFlag_SEIS (RNG_TypeDef * RNGx)

Function description

Indicate if the Seed Error Interrupt Status Flag is set or not.

Parameters

- **RNGx:** RNG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR SEIS LL_RNG_IsActiveFlag_SEIS

LL_RNG_ClearFlag_CEIS

Function name

__STATIC_INLINE void LL_RNG_ClearFlag_CEIS (RNG_TypeDef * RNGx)

Function description

Clear Clock Error interrupt Status (CEIS) Flag.

Parameters

- **RNGx:** RNG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CEIS LL_RNG_ClearFlag_CEIS

LL_RNG_ClearFlag_SEIS

Function name

__STATIC_INLINE void LL_RNG_ClearFlag_SEIS (RNG_TypeDef * RNGx)

Function description

Clear Seed Error interrupt Status (SEIS) Flag.

Parameters

- **RNGx:** RNG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR SEIS LL_RNG_ClearFlag_SEIS

LL_RNG_EnableIT

Function name

__STATIC_INLINE void LL_RNG_EnableIT (RNG_TypeDef * RNGx)

Function description

Enable Random Number Generator Interrupt (applies for either Seed error, Clock Error or Data ready interrupts)

Parameters

- **RNGx:** RNG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR IE LL_RNG_EnableIT

LL_RNG_DisableIT

Function name

__STATIC_INLINE void LL_RNG_DisableIT (RNG_TypeDef * RNGx)

Function description

Disable Random Number Generator Interrupt (applies for either Seed error, Clock Error or Data ready interrupts)

Parameters

- **RNGx:** RNG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR IE LL_RNG_DisableIT

LL_RNG_IsEnabledIT

Function name

__STATIC_INLINE uint32_t LL_RNG_IsEnabledIT (RNG_TypeDef * RNGx)

Function description

Check if Random Number Generator Interrupt is enabled (applies for either Seed error, Clock Error or Data ready interrupts)

Parameters

- **RNGx:** RNG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR IE LL_RNG_IsEnabledIT

LL_RNG_ReadRandData32

Function name

__STATIC_INLINE uint32_t LL_RNG_ReadRandData32 (RNG_TypeDef * RNGx)

Function description

Return32-bit Random Number value.

Parameters

- **RNGx:** RNG Instance

Return values

- **Generated:** 32-bit random value

Reference Manual to LL API cross reference:

- DR RNDATA LL_RNG_ReadRandData32

LL_RNG_DeInit

Function name

ErrorStatus LL_RNG_DeInit (RNG_TypeDef * RNGx)

Function description

De-initialize RNG registers (Registers restored to their default values).

Parameters

- **RNGx:** RNG Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RNG registers are de-initialized
  - ERROR: not applicable

88.2 RNG Firmware driver defines

The following section lists the various define and macros of the module.

88.2.1 RNG

RNG

Get Flags Defines

LL_RNG_SR_DRDY

Register contains valid random data

LL_RNG_SR_CECS

Clock error current status

LL_RNG_SR_SECS

Seed error current status

LL_RNG_SR_CEIS

Clock error interrupt status

LL_RNG_SR_SEIS

Seed error interrupt status

IT Defines

LL_RNG_CR_IE

RNG Interrupt enable

Common Write and read registers Macros

LL_RNG_WriteReg

Description:

- Write a value in RNG register.

Parameters:

- __INSTANCE__: RNG Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_RNG_ReadReg

Description:

- Read a value in RNG register.

Parameters:

- __INSTANCE__: RNG Instance
- __REG__: Register to be read

Return value:

- Register: value

89 LL RTC Generic Driver

89.1 RTC Firmware driver registers structures

89.1.1 LL_RTC_InitTypeDef

LL_RTC_InitTypeDef is defined in the stm32f4xx_ll_rtc.h

Data Fields

- ***uint32_t HourFormat***
- ***uint32_t AsynchPrescaler***
- ***uint32_t SynchPrescaler***

Field Documentation

- ***uint32_t LL_RTC_InitTypeDef::HourFormat***

  Specifies the RTC Hours Format. This parameter can be a value of ***RTC_LL_EC_HOURFORMAT***This feature can be modified afterwards using unitary function **LL_RTC_SetHourFormat()**.
- ***uint32_t LL_RTC_InitTypeDef::AsynchPrescaler***

  Specifies the RTC Asynchronous Predivider value. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x7FThis feature can be modified afterwards using unitary function **LL_RTC_SetAsynchPrescaler()**.
- ***uint32_t LL_RTC_InitTypeDef::SynchPrescaler***

  Specifies the RTC Synchronous Predivider value. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0x7FFFThis feature can be modified afterwards using unitary function **LL_RTC_SetSynchPrescaler()**.

89.1.2 LL_RTC_TimeTypeDef

LL_RTC_TimeTypeDef is defined in the stm32f4xx_ll_rtc.h

Data Fields

- ***uint32_t TimeFormat***
- ***uint8_t Hours***
- ***uint8_t Minutes***
- ***uint8_t Seconds***

Field Documentation

- ***uint32_t LL_RTC_TimeTypeDef::TimeFormat***

  Specifies the RTC AM/PM Time. This parameter can be a value of ***RTC_LL_EC_TIME_FORMAT***This feature can be modified afterwards using unitary function **LL_RTC_TIME_SetFormat()**.
- ***uint8_t LL_RTC_TimeTypeDef::Hours***

  Specifies the RTC Time Hours. This parameter must be a number between Min_Data = 0 and Max_Data = 12 if the **LL_RTC_TIME_FORMAT_PM** is selected. This parameter must be a number between Min_Data = 0 and Max_Data = 23 if the **LL_RTC_TIME_FORMAT_AM_OR_24** is selected.This feature can be modified afterwards using unitary function **LL_RTC_TIME_SetHour()**.
- ***uint8_t LL_RTC_TimeTypeDef::Minutes***

  Specifies the RTC Time Minutes. This parameter must be a number between Min_Data = 0 and Max_Data = 59This feature can be modified afterwards using unitary function **LL_RTC_TIME_SetMinute()**.
- ***uint8_t LL_RTC_TimeTypeDef::Seconds***

  Specifies the RTC Time Seconds. This parameter must be a number between Min_Data = 0 and Max_Data = 59This feature can be modified afterwards using unitary function **LL_RTC_TIME_SetSecond()**.

89.1.3 LL_RTC_DateTypeDef

LL_RTC_DateTypeDef is defined in the stm32f4xx_ll_rtc.h

Data Fields

- ***uint8_t WeekDay***
- ***uint8_t Month***

- ***uint8_t Day***
- ***uint8_t Year***

Field Documentation

- ***uint8_t LL_RTC_DateTypeDef::WeekDay***

  Specifies the RTC Date WeekDay. This parameter can be a value of ***RTC_LL_EC_WEEKDAY***This feature can be modified afterwards using unitary function **LL_RTC_DATE_SetWeekDay()**.
- ***uint8_t LL_RTC_DateTypeDef::Month***

  Specifies the RTC Date Month. This parameter can be a value of ***RTC_LL_EC_MONTH***This feature can be modified afterwards using unitary function **LL_RTC_DATE_SetMonth()**.
- ***uint8_t LL_RTC_DateTypeDef::Day***

  Specifies the RTC Date Day. This parameter must be a number between Min_Data = 1 and Max_Data = 31This feature can be modified afterwards using unitary function **LL_RTC_DATE_SetDay()**.
- ***uint8_t LL_RTC_DateTypeDef::Year***

  Specifies the RTC Date Year. This parameter must be a number between Min_Data = 0 and Max_Data = 99This feature can be modified afterwards using unitary function **LL_RTC_DATE_SetYear()**.

89.1.4 LL_RTC_AlarmTypeDef

LL_RTC_AlarmTypeDef is defined in the stm32f4xx_ll_rtc.h

Data Fields

- ***LL_RTC_TimeTypeDef AlarmTime***
- ***uint32_t AlarmMask***
- ***uint32_t AlarmDateWeekDaySel***
- ***uint8_t AlarmDateWeekDay***

Field Documentation

- ***LL_RTC_TimeTypeDef LL_RTC_AlarmTypeDef::AlarmTime***

  Specifies the RTC Alarm Time members.
- ***uint32_t LL_RTC_AlarmTypeDef::AlarmMask***

  Specifies the RTC Alarm Masks. This parameter can be a value of ***RTC_LL_EC_ALMA_MASK*** for ALARM A or ***RTC_LL_EC_ALMB_MASK*** for ALARM B.This feature can be modified afterwards using unitary function **LL_RTC_ALMA_SetMask()** for ALARM A or **LL_RTC_ALMB_SetMask()** for ALARM B
- ***uint32_t LL_RTC_AlarmTypeDef::AlarmDateWeekDaySel***

  Specifies the RTC Alarm is on day or WeekDay. This parameter can be a value of ***RTC_LL_EC_ALMA_WEEKDAY_SELECTION*** for ALARM A or ***RTC_LL_EC_ALMB_WEEKDAY_SELECTION*** for ALARM BThis feature can be modified afterwards using unitary function **LL_RTC_ALMA_EnableWeekday()** or **LL_RTC_ALMA_DisableWeekday()** for ALARM A or **LL_RTC_ALMB_EnableWeekday()** or **LL_RTC_ALMB_DisableWeekday()** for ALARM B
- ***uint8_t LL_RTC_AlarmTypeDef::AlarmDateWeekDay***

  Specifies the RTC Alarm Day/WeekDay. If AlarmDateWeekDaySel set to day, this parameter must be a number between Min_Data = 1 and Max_Data = 31.This feature can be modified afterwards using unitary function **LL_RTC_ALMA_SetDay()** for ALARM A or **LL_RTC_ALMB_SetDay()** for ALARM B.If AlarmDateWeekDaySel set to Weekday, this parameter can be a value of ***RTC_LL_EC_WEEKDAY***.This feature can be modified afterwards using unitary function **LL_RTC_ALMA_SetWeekDay()** for ALARM A or **LL_RTC_ALMB_SetWeekDay()** for ALARM B.

89.2 RTC Firmware driver API description

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

89.2.1 Detailed description of functions

LL_RTC_SetHourFormat

Function name

__STATIC_INLINE void LL_RTC_SetHourFormat (RTC_TypeDef * RTCx, uint32_t HourFormat)

Function description

Set Hours format (24 hour/day or AM/PM hour format)

Parameters

- **RTCx:** RTC Instance
- **HourFormat:** This parameter can be one of the following values:
  - LL_RTC_HOURFORMAT_24HOUR
  - LL_RTC_HOURFORMAT_AMPM

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)

Reference Manual to LL API cross reference:

- CR FMT LL_RTC_SetHourFormat

LL_RTC_GetHourFormat

Function name

__STATIC_INLINE uint32_t LL_RTC_GetHourFormat (RTC_TypeDef * RTCx)

Function description

Get Hours format (24 hour/day or AM/PM hour format)

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_HOURFORMAT_24HOUR
  - LL_RTC_HOURFORMAT_AMPM

Reference Manual to LL API cross reference:

- CR FMT LL_RTC_GetHourFormat

LL_RTC_SetAlarmOutEvent

Function name

__STATIC_INLINE void LL_RTC_SetAlarmOutEvent (RTC_TypeDef * RTCx, uint32_t AlarmOutput)

Function description

Select the flag to be routed to RTC_ALARM output.

Parameters

- **RTCx:** RTC Instance
- **AlarmOutput:** This parameter can be one of the following values:
  - LL_RTC_ALARMOUT_DISABLE
  - LL_RTC_ALARMOUT_ALMA
  - LL_RTC_ALARMOUT_ALMB
  - LL_RTC_ALARMOUT_WAKEUP

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR OSEL LL_RTC_SetAlarmOutEvent

LL_RTC_GetAlarmOutEvent

Function name

__STATIC_INLINE uint32_t LL_RTC_GetAlarmOutEvent (RTC_TypeDef * RTCx)

Function description

Get the flag to be routed to RTC_ALARM output.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_ALARMOUT_DISABLE
  - LL_RTC_ALARMOUT_ALMA
  - LL_RTC_ALARMOUT_ALMB
  - LL_RTC_ALARMOUT_WAKEUP

Reference Manual to LL API cross reference:

- CR OSEL LL_RTC_GetAlarmOutEvent

LL_RTC_SetAlarmOutputType

Function name

__STATIC_INLINE void LL_RTC_SetAlarmOutputType (RTC_TypeDef * RTCx, uint32_t Output)

Function description

Set RTC_ALARM output type (ALARM in push-pull or open-drain output)

Parameters

- **RTCx:** RTC Instance
- **Output:** This parameter can be one of the following values:
  - LL_RTC_ALARM_OUTPUTTYPE_OPENDRAIN
  - LL_RTC_ALARM_OUTPUTTYPE_PUSHPULL

Return values

- **None:**

Notes

- Used only when RTC_ALARM is mapped on PC13
- If all RTC alternate functions are disabled and PC13MODE = 1, PC13VALUE configures the PC13 output data

Reference Manual to LL API cross reference:

- TAFCR ALARMOUTTYPE LL_RTC_SetAlarmOutputType

LL_RTC_GetAlarmOutputType

Function name

__STATIC_INLINE uint32_t LL_RTC_GetAlarmOutputType (RTC_TypeDef * RTCx)

Function description

Get RTC_ALARM output type (ALARM in push-pull or open-drain output)

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_ALARM_OUTPUTTYPE_OPENDRAIN
  - LL_RTC_ALARM_OUTPUTTYPE_PUSHPULL

Notes

- used only when RTC_ALARM is mapped on PC13
- If all RTC alternate functions are disabled and PC13MODE = 1, PC13VALUE configures the PC13 output data

Reference Manual to LL API cross reference:

- TAFCR ALARMOUTTYPE LL_RTC_GetAlarmOutputType

LL_RTC_EnablePushPullMode

Function name

__STATIC_INLINE void LL_RTC_EnablePushPullMode (RTC_TypeDef * RTCx, uint32_t PinMask)

Function description

Enable push-pull output on PC13, PC14 and/or PC15.

Parameters

- **RTCx:** RTC Instance
- **PinMask:** This parameter can be a combination of the following values:
  - LL_RTC_PIN_PC13
  - LL_RTC_PIN_PC14
  - LL_RTC_PIN_PC15

Return values

- **None:**

Notes

- PC13 forced to push-pull output if all RTC alternate functions are disabled
- PC14 and PC15 forced to push-pull output if LSE is disabled

Reference Manual to LL API cross reference:

- TAFCR PC13MODE LL_RTC_EnablePushPullMode
-
- TAFCR PC14MODE LL_RTC_EnablePushPullMode
-
- TAFCR PC15MODE LL_RTC_EnablePushPullMode

LL_RTC_DisablePushPullMode

Function name

__STATIC_INLINE void LL_RTC_DisablePushPullMode (RTC_TypeDef * RTCx, uint32_t PinMask)

Function description

Disable push-pull output on PC13, PC14 and/or PC15.

Parameters

- **RTCx:** RTC Instance
- **PinMask:** This parameter can be a combination of the following values:
  - LL_RTC_PIN_PC13
  - LL_RTC_PIN_PC14
  - LL_RTC_PIN_PC15

Return values

- **None:**

Notes

- PC13, PC14 and/or PC15 are controlled by the GPIO configuration registers. Consequently PC13, PC14 and/or PC15 are floating in Standby mode.

Reference Manual to LL API cross reference:

- TAFCR PC13MODE LL_RTC_DisablePushPullMode
- TAFCR PC14MODE LL_RTC_DisablePushPullMode
- TAFCR PC15MODE LL_RTC_DisablePushPullMode

LL_RTC_SetOutputPin

Function name

__STATIC_INLINE void LL_RTC_SetOutputPin (RTC_TypeDef * RTCx, uint32_t PinMask)

Function description

Set PC14 and/or PC15 to high level.

Parameters

- **RTCx:** RTC Instance
- **PinMask:** This parameter can be a combination of the following values:
  - LL_RTC_PIN_PC14
  - LL_RTC_PIN_PC15

Return values

- **None:**

Notes

- Output data configuration is possible if the LSE is disabled and PushPull output is enabled (through LL_RTC_EnablePushPullMode)

Reference Manual to LL API cross reference:

- TAFCR PC14VALUE LL_RTC_SetOutputPin
- TAFCR PC15VALUE LL_RTC_SetOutputPin

LL_RTC_ResetOutputPin

Function name

__STATIC_INLINE void LL_RTC_ResetOutputPin (RTC_TypeDef * RTCx, uint32_t PinMask)

Function description

Set PC14 and/or PC15 to low level.

Parameters

- **RTCx:** RTC Instance
- **PinMask:** This parameter can be a combination of the following values:
  - LL_RTC_PIN_PC14
  - LL_RTC_PIN_PC15

Return values

- **None:**

Notes

- Output data configuration is possible if the LSE is disabled and PushPull output is enabled (through LL_RTC_EnablePushPullMode)

Reference Manual to LL API cross reference:

- TAFCR PC14VALUE LL_RTC_ResetOutputPin
- TAFCR PC15VALUE LL_RTC_ResetOutputPin

LL_RTC_EnableInitMode

Function name

__STATIC_INLINE void LL_RTC_EnableInitMode (RTC_TypeDef * RTCx)

Function description

Enable initialization mode.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Initialization mode is used to program time and date register (RTC_TR and RTC_DR) and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Reference Manual to LL API cross reference:

- ISR INIT LL_RTC_EnableInitMode

LL_RTC_DisableInitMode

Function name

__STATIC_INLINE void LL_RTC_DisableInitMode (RTC_TypeDef * RTCx)

Function description

Disable initialization mode (Free running mode)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR INIT LL_RTC_DisableInitMode

LL_RTC_SetOutputPolarity

Function name

__STATIC_INLINE void LL_RTC_SetOutputPolarity (RTC_TypeDef * RTCx, uint32_t Polarity)

Function description

Set Output polarity (pin is low when ALRAF/ALRBF/WUTF is asserted)

Parameters

- **RTCx:** RTC Instance
- **Polarity:** This parameter can be one of the following values:
  - LL_RTC_OUTPUTPOLARITY_PIN_HIGH
  - LL_RTC_OUTPUTPOLARITY_PIN_LOW

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR POL LL_RTC_SetOutputPolarity

LL_RTC_GetOutputPolarity

Function name

__STATIC_INLINE uint32_t LL_RTC_GetOutputPolarity (RTC_TypeDef * RTCx)

Function description

Get Output polarity.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_OUTPUTPOLARITY_PIN_HIGH
  - LL_RTC_OUTPUTPOLARITY_PIN_LOW

Reference Manual to LL API cross reference:

- CR POL LL_RTC_GetOutputPolarity

LL_RTC_EnableShadowRegBypass

Function name

__STATIC_INLINE void LL_RTC_EnableShadowRegBypass (RTC_TypeDef * RTCx)

Function description

Enable Bypass the shadow registers.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR BYPSHAD LL_RTC_EnableShadowRegBypass

LL_RTC_DisableShadowRegBypass

Function name

__STATIC_INLINE void LL_RTC_DisableShadowRegBypass (RTC_TypeDef * RTCx)

Function description

Disable Bypass the shadow registers.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR BYPSHAD LL_RTC_DisableShadowRegBypass

LL_RTC_IsShadowRegBypassEnabled

Function name

__STATIC_INLINE uint32_t LL_RTC_IsShadowRegBypassEnabled (RTC_TypeDef * RTCx)

Function description

Check if Shadow registers bypass is enabled or not.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR BYPSHAD LL_RTC_IsShadowRegBypassEnabled

LL_RTC_EnableRefClock

Function name

__STATIC_INLINE void LL_RTC_EnableRefClock (RTC_TypeDef * RTCx)

Function description

Enable RTC_REFIN reference clock detection (50 or 60 Hz)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)

Reference Manual to LL API cross reference:

- CR REFCKON LL_RTC_EnableRefClock

LL_RTC_DisableRefClock

Function name

__STATIC_INLINE void LL_RTC_DisableRefClock (RTC_TypeDef * RTCx)

Function description

Disable RTC_REFIN reference clock detection (50 or 60 Hz)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)

Reference Manual to LL API cross reference:

- CR REFCKON LL_RTC_DisableRefClock

LL_RTC_SetAsynchPrescaler

Function name

__STATIC_INLINE void LL_RTC_SetAsynchPrescaler (RTC_TypeDef * RTCx, uint32_t AsynchPrescaler)

Function description

Set Asynchronous prescaler factor.

Parameters

- **RTCx:** RTC Instance
- **AsynchPrescaler:** Value between Min_Data = 0 and Max_Data = 0x7F

Return values

- **None:**

Reference Manual to LL API cross reference:

- PRER PREDIV_A LL_RTC_SetAsynchPrescaler

LL_RTC_SetSynchPrescaler

Function name

__STATIC_INLINE void LL_RTC_SetSynchPrescaler (RTC_TypeDef * RTCx, uint32_t SynchPrescaler)

Function description

Set Synchronous prescaler factor.

Parameters

- **RTCx:** RTC Instance
- **SynchPrescaler:** Value between Min_Data = 0 and Max_Data = 0x7FFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- PRER PREDIV_S LL_RTC_SetSynchPrescaler

LL_RTC_GetAsynchPrescaler

Function name

__STATIC_INLINE uint32_t LL_RTC_GetAsynchPrescaler (RTC_TypeDef * RTCx)

Function description

Get Asynchronous prescaler factor.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data = 0 and Max_Data = 0x7F

Reference Manual to LL API cross reference:

- PRER PREDIV_A LL_RTC_GetAsynchPrescaler

LL_RTC_GetSynchPrescaler

Function name

__STATIC_INLINE uint32_t LL_RTC_GetSynchPrescaler (RTC_TypeDef * RTCx)

Function description

Get Synchronous prescaler factor.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data = 0 and Max_Data = 0x7FFF

Reference Manual to LL API cross reference:

- PRER PREDIV_S LL_RTC_GetSynchPrescaler

LL_RTC_EnableWriteProtection

Function name

__STATIC_INLINE void LL_RTC_EnableWriteProtection (RTC_TypeDef * RTCx)

Function description

Enable the write protection for RTC registers.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- WPR KEY LL_RTC_EnableWriteProtection

LL_RTC_DisableWriteProtection

Function name

__STATIC_INLINE void LL_RTC_DisableWriteProtection (RTC_TypeDef * RTCx)

Function description

Disable the write protection for RTC registers.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- WPR KEY LL_RTC_DisableWriteProtection

LL_RTC_TIME_SetFormat

Function name

__STATIC_INLINE void LL_RTC_TIME_SetFormat (RTC_TypeDef * RTCx, uint32_t TimeFormat)

Function description

Set time format (AM/24-hour or PM notation)

Parameters

- **RTCx:** RTC Instance
- **TimeFormat:** This parameter can be one of the following values:
  - LL_RTC_TIME_FORMAT_AM_OR_24
  - LL_RTC_TIME_FORMAT_PM

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)

Reference Manual to LL API cross reference:

- TR PM LL_RTC_TIME_SetFormat

LL_RTC_TIME_GetFormat

Function name

__STATIC_INLINE uint32_t LL_RTC_TIME_GetFormat (RTC_TypeDef * RTCx)

Function description

Get time format (AM or PM notation)

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_TIME_FORMAT_AM_OR_24
  - LL_RTC_TIME_FORMAT_PM

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)).

Reference Manual to LL API cross reference:

- TR PM LL_RTC_TIME_GetFormat

LL_RTC_TIME_SetHour

Function name

__STATIC_INLINE void LL_RTC_TIME_SetHour (RTC_TypeDef * RTCx, uint32_t Hours)

Function description

Set Hours in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Hours:** Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)
- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert hour from binary to BCD format

Reference Manual to LL API cross reference:

- TR HT LL_RTC_TIME_SetHour
- TR HU LL_RTC_TIME_SetHour

LL_RTC_TIME_GetHour

Function name

__STATIC_INLINE uint32_t LL_RTC_TIME_GetHour (RTC_TypeDef * RTCx)

Function description

Get Hours in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)).
- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert hour from BCD to Binary format

Reference Manual to LL API cross reference:

- TR HT LL_RTC_TIME_GetHour
- TR HU LL_RTC_TIME_GetHour

LL_RTC_TIME_SetMinute

Function name

__STATIC_INLINE void LL_RTC_TIME_SetMinute (RTC_TypeDef * RTCx, uint32_t Minutes)

Function description

Set Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Minutes:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)
- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Minutes from binary to BCD format

Reference Manual to LL API cross reference:

- TR MNT LL_RTC_TIME_SetMinute
- TR MNU LL_RTC_TIME_SetMinute

LL_RTC_TIME_GetMinute

Function name

__STATIC_INLINE uint32_t LL_RTC_TIME_GetMinute (RTC_TypeDef * RTCx)

Function description

Get Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)).
- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert minute from BCD to Binary format

Reference Manual to LL API cross reference:

- TR MNT LL_RTC_TIME_GetMinute
- TR MNU LL_RTC_TIME_GetMinute

LL_RTC_TIME_SetSecond

Function name

__STATIC_INLINE void LL_RTC_TIME_SetSecond (RTC_TypeDef * RTCx, uint32_t Seconds)

Function description

Set Seconds in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Seconds:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)
- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Seconds from binary to BCD format

Reference Manual to LL API cross reference:

- TR ST LL_RTC_TIME_SetSecond
- TR SU LL_RTC_TIME_SetSecond

LL_RTC_TIME_GetSecond

Function name

__STATIC_INLINE uint32_t LL_RTC_TIME_GetSecond (RTC_TypeDef * RTCx)

Function description

Get Seconds in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)).
- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Seconds from BCD to Binary format

Reference Manual to LL API cross reference:

- TR ST LL_RTC_TIME_GetSecond
- TR SU LL_RTC_TIME_GetSecond

LL_RTC_TIME_Config

Function name

__STATIC_INLINE void LL_RTC_TIME_Config (RTC_TypeDef * RTCx, uint32_t Format12_24, uint32_t Hours, uint32_t Minutes, uint32_t Seconds)

Function description

Set time (hour, minute and second) in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Format12_24:** This parameter can be one of the following values:
  - LL_RTC_TIME_FORMAT_AM_OR_24
  - LL_RTC_TIME_FORMAT_PM
- **Hours:** Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23
- **Minutes:** Value between Min_Data=0x00 and Max_Data=0x59
- **Seconds:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)
- TimeFormat and Hours should follow the same format

Reference Manual to LL API cross reference:

- TR PM LL_RTC_TIME_Config
- TR HT LL_RTC_TIME_Config
- TR HU LL_RTC_TIME_Config
- TR MNT LL_RTC_TIME_Config
- TR MNU LL_RTC_TIME_Config
- TR ST LL_RTC_TIME_Config
- TR SU LL_RTC_TIME_Config

LL_RTC_TIME_Get

Function name

__STATIC_INLINE uint32_t LL_RTC_TIME_Get (RTC_TypeDef * RTCx)

Function description

Get time (hour, minute and second) in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Combination:** of hours, minutes and seconds (Format: 0x00HHMMSS).

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- Read either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read (LL_RTC_ReadReg(RTC, DR)).
- helper macros __LL_RTC_GET_HOUR, __LL_RTC_GET_MINUTE and __LL_RTC_GET_SECOND are available to get independently each parameter.

Reference Manual to LL API cross reference:

- TR HT LL_RTC_TIME_Get
- TR HU LL_RTC_TIME_Get
- TR MNT LL_RTC_TIME_Get
- TR MNU LL_RTC_TIME_Get
- TR ST LL_RTC_TIME_Get
- TR SU LL_RTC_TIME_Get

LL_RTC_TIME_EnableDayLightStore

Function name

__STATIC_INLINE void LL_RTC_TIME_EnableDayLightStore (RTC_TypeDef * RTCx)

Function description

Memorize whether the daylight saving time change has been performed.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR BKP LL_RTC_TIME_EnableDayLightStore

LL_RTC_TIME_DisableDayLightStore

Function name

__STATIC_INLINE void LL_RTC_TIME_DisableDayLightStore (RTC_TypeDef * RTCx)

Function description

Disable memorization whether the daylight saving time change has been performed.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR BKP LL_RTC_TIME_DisableDayLightStore

LL_RTC_TIME_IsDayLightStoreEnabled

Function name

__STATIC_INLINE uint32_t LL_RTC_TIME_IsDayLightStoreEnabled (RTC_TypeDef * RTCx)

Function description

Check if RTC Day Light Saving stored operation has been enabled or not.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR BKP LL_RTC_TIME_IsDayLightStoreEnabled

LL_RTC_TIME_DecHour

Function name

__STATIC_INLINE void LL_RTC_TIME_DecHour (RTC_TypeDef * RTCx)

Function description

Subtract 1 hour (winter time change)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR SUB1H LL_RTC_TIME_DecHour

LL_RTC_TIME_IncHour

Function name

__STATIC_INLINE void LL_RTC_TIME_IncHour (RTC_TypeDef * RTCx)

Function description

Add 1 hour (summer time change)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ADD1H LL_RTC_TIME_IncHour

LL_RTC_TIME_GetSubSecond

Function name

__STATIC_INLINE uint32_t LL_RTC_TIME_GetSubSecond (RTC_TypeDef * RTCx)

Function description

Get Sub second value in the synchronous prescaler counter.

Parameters

- **RTCx:** RTC Instance

Return values

- **Sub:** second value (number between 0 and 65535)

Notes

- You can use both SubSeconds value and SecondFraction (PREDIV_S through LL_RTC_GetSynchPrescaler function) terms returned to convert Calendar SubSeconds value in second fraction ratio with time unit following generic formula: ==> Seconds fraction ratio * time_unit= [(SecondFraction-SubSeconds)/(SecondFraction+1)] * time_unit This conversion can be performed only if no shift operation is pending (ie. SHFP=0) when PREDIV_S >= SS.

Reference Manual to LL API cross reference:

- SSR SS LL_RTC_TIME_GetSubSecond

LL_RTC_TIME_Synchronize

Function name

__STATIC_INLINE void LL_RTC_TIME_Synchronize (RTC_TypeDef * RTCx, uint32_t ShiftSecond, uint32_t Fraction)

Function description

Synchronize to a remote clock with a high degree of precision.

Parameters

- **RTCx:** RTC Instance
- **ShiftSecond:** This parameter can be one of the following values:
  - LL_RTC_SHIFT_SECOND_DELAY
  - LL_RTC_SHIFT_SECOND_ADVANCE
- **Fraction:** Number of Seconds Fractions (any value from 0 to 0x7FFF)

Return values

- **None:**

Notes

- This operation effectively subtracts from (delays) or advance the clock of a fraction of a second.
- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- When REFCKON is set, firmware must not write to Shift control register.

Reference Manual to LL API cross reference:

- SHIFTR ADD1S LL_RTC_TIME_Synchronize
- SHIFTR SUBFS LL_RTC_TIME_Synchronize

LL_RTC_DATE_SetYear

Function name

__STATIC_INLINE void LL_RTC_DATE_SetYear (RTC_TypeDef * RTCx, uint32_t Year)

Function description

Set Year in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Year:** Value between Min_Data=0x00 and Max_Data=0x99

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Year from binary to BCD format

Reference Manual to LL API cross reference:

- DR YT LL_RTC_DATE_SetYear
- DR YU LL_RTC_DATE_SetYear

LL_RTC_DATE_GetYear

Function name

__STATIC_INLINE uint32_t LL_RTC_DATE_GetYear (RTC_TypeDef * RTCx)

Function description

Get Year in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x99

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Year from BCD to Binary format

Reference Manual to LL API cross reference:

- DR YT LL_RTC_DATE_GetYear
- DR YU LL_RTC_DATE_GetYear

LL_RTC_DATE_SetWeekDay

Function name

__STATIC_INLINE void LL_RTC_DATE_SetWeekDay (RTC_TypeDef * RTCx, uint32_t WeekDay)

Function description

Set Week day.

Parameters

- **RTCx:** RTC Instance
- **WeekDay:** This parameter can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR WDU LL_RTC_DATE_SetWeekDay

LL_RTC_DATE_GetWeekDay

Function name

__STATIC_INLINE uint32_t LL_RTC_DATE_GetWeekDay (RTC_TypeDef * RTCx)

Function description

Get Week day.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit

Reference Manual to LL API cross reference:

- DR WDU LL_RTC_DATE_GetWeekDay

LL_RTC_DATE_SetMonth

Function name

__STATIC_INLINE void LL_RTC_DATE_SetMonth (RTC_TypeDef * RTCx, uint32_t Month)

Function description

Set Month in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Month:** This parameter can be one of the following values:
  - LL_RTC_MONTH_JANUARY
  - LL_RTC_MONTH_FEBRUARY
  - LL_RTC_MONTH_MARCH
  - LL_RTC_MONTH_APRIL
  - LL_RTC_MONTH_MAY
  - LL_RTC_MONTH_JUNE
  - LL_RTC_MONTH_JULY
  - LL_RTC_MONTH_AUGUST
  - LL_RTC_MONTH_SEPTEMBER
  - LL_RTC_MONTH_OCTOBER
  - LL_RTC_MONTH_NOVEMBER
  - LL_RTC_MONTH_DECEMBER

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Month from binary to BCD format

Reference Manual to LL API cross reference:

- DR MT LL_RTC_DATE_SetMonth
- DR MU LL_RTC_DATE_SetMonth

LL_RTC_DATE_GetMonth

Function name

__STATIC_INLINE uint32_t LL_RTC_DATE_GetMonth (RTC_TypeDef * RTCx)

Function description

Get Month in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_MONTH_JANUARY
  - LL_RTC_MONTH_FEBRUARY
  - LL_RTC_MONTH_MARCH
  - LL_RTC_MONTH_APRIL
  - LL_RTC_MONTH_MAY
  - LL_RTC_MONTH_JUNE
  - LL_RTC_MONTH_JULY
  - LL_RTC_MONTH_AUGUST
  - LL_RTC_MONTH_SEPTEMBER
  - LL_RTC_MONTH_OCTOBER
  - LL_RTC_MONTH_NOVEMBER
  - LL_RTC_MONTH_DECEMBER

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Month from BCD to Binary format

Reference Manual to LL API cross reference:

- DR MT LL_RTC_DATE_GetMonth
- DR MU LL_RTC_DATE_GetMonth

LL_RTC_DATE_SetDay

Function name

__STATIC_INLINE void LL_RTC_DATE_SetDay (RTC_TypeDef * RTCx, uint32_t Day)

Function description

Set Day in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Day:** Value between Min_Data=0x01 and Max_Data=0x31

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Day from binary to BCD format

Reference Manual to LL API cross reference:

- DR DT LL_RTC_DATE_SetDay
- DR DU LL_RTC_DATE_SetDay

LL_RTC_DATE_GetDay

Function name

__STATIC_INLINE uint32_t LL_RTC_DATE_GetDay (RTC_TypeDef * RTCx)

Function description

Get Day in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x31

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Day from BCD to Binary format

Reference Manual to LL API cross reference:

- DR DT LL_RTC_DATE_GetDay
- DR DU LL_RTC_DATE_GetDay

LL_RTC_DATE_Config

Function name

__STATIC_INLINE void LL_RTC_DATE_Config (RTC_TypeDef * RTCx, uint32_t WeekDay, uint32_t Day, uint32_t Month, uint32_t Year)

Function description

Set date (WeekDay, Day, Month and Year) in BCD format.

Parameters

- **RTCx:** RTC Instance
- **WeekDay:** This parameter can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY
- **Day:** Value between Min_Data=0x01 and Max_Data=0x31
- **Month:** This parameter can be one of the following values:
  - LL_RTC_MONTH_JANUARY
  - LL_RTC_MONTH_FEBRUARY
  - LL_RTC_MONTH_MARCH
  - LL_RTC_MONTH_APRIL
  - LL_RTC_MONTH_MAY
  - LL_RTC_MONTH_JUNE
  - LL_RTC_MONTH_JULY
  - LL_RTC_MONTH_AUGUST
  - LL_RTC_MONTH_SEPTEMBER
  - LL_RTC_MONTH_OCTOBER
  - LL_RTC_MONTH_NOVEMBER
  - LL_RTC_MONTH_DECEMBER
- **Year:** Value between Min_Data=0x00 and Max_Data=0x99

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR WDU LL_RTC_DATE_Config
- DR MT LL_RTC_DATE_Config
- DR MU LL_RTC_DATE_Config
- DR DT LL_RTC_DATE_Config
- DR DU LL_RTC_DATE_Config
- DR YT LL_RTC_DATE_Config
- DR YU LL_RTC_DATE_Config

LL_RTC_DATE_Get

Function name

__STATIC_INLINE uint32_t LL_RTC_DATE_Get (RTC_TypeDef * RTCx)

Function description

Get date (WeekDay, Day, Month and Year) in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Combination:** of WeekDay, Day, Month and Year (Format: 0xWWDDMMYY).

Notes

- if shadow mode is disabled (BYPSHAD=0), need to check if RSF flag is set before reading this bit
- helper macros __LL_RTC_GET_WEEKDAY, __LL_RTC_GET_YEAR, __LL_RTC_GET_MONTH, and __LL_RTC_GET_DAY are available to get independently each parameter.

Reference Manual to LL API cross reference:

- DR WDU LL_RTC_DATE_Get
- DR MT LL_RTC_DATE_Get
- DR MU LL_RTC_DATE_Get
- DR DT LL_RTC_DATE_Get
- DR DU LL_RTC_DATE_Get
- DR YT LL_RTC_DATE_Get
- DR YU LL_RTC_DATE_Get

LL_RTC_ALMA_Enable

Function name

__STATIC_INLINE void LL_RTC_ALMA_Enable (RTC_TypeDef * RTCx)

Function description

Enable Alarm A.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ALRAE LL_RTC_ALMA_Enable

LL_RTC_ALMA_Disable

Function name

__STATIC_INLINE void LL_RTC_ALMA_Disable (RTC_TypeDef * RTCx)

Function description

Disable Alarm A.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ALRAE LL_RTC_ALMA_Disable

LL_RTC_ALMA_SetMask

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetMask (RTC_TypeDef * RTCx, uint32_t Mask)

Function description

Specify the Alarm A masks.

Parameters

- **RTCx:** RTC Instance
- **Mask:** This parameter can be a combination of the following values:
  - LL_RTC_ALMA_MASK_NONE
  - LL_RTC_ALMA_MASK_DATEWEEKDAY
  - LL_RTC_ALMA_MASK_HOURS
  - LL_RTC_ALMA_MASK_MINUTES
  - LL_RTC_ALMA_MASK_SECONDS
  - LL_RTC_ALMA_MASK_ALL

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMAR MSK4 LL_RTC_ALMA_SetMask
- ALRMAR MSK3 LL_RTC_ALMA_SetMask
- ALRMAR MSK2 LL_RTC_ALMA_SetMask
- ALRMAR MSK1 LL_RTC_ALMA_SetMask

LL_RTC_ALMA_GetMask

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetMask (RTC_TypeDef * RTCx)

Function description

Get the Alarm A masks.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be can be a combination of the following values:
  - LL_RTC_ALMA_MASK_NONE
  - LL_RTC_ALMA_MASK_DATEWEEKDAY
  - LL_RTC_ALMA_MASK_HOURS
  - LL_RTC_ALMA_MASK_MINUTES
  - LL_RTC_ALMA_MASK_SECONDS
  - LL_RTC_ALMA_MASK_ALL

Reference Manual to LL API cross reference:

- ALRMAR MSK4 LL_RTC_ALMA_GetMask
- ALRMAR MSK3 LL_RTC_ALMA_GetMask
- ALRMAR MSK2 LL_RTC_ALMA_GetMask
- ALRMAR MSK1 LL_RTC_ALMA_GetMask

LL_RTC_ALMA_EnableWeekday

Function name

__STATIC_INLINE void LL_RTC_ALMA_EnableWeekday (RTC_TypeDef * RTCx)

Function description

Enable AlarmA Week day selection (DU[3:0] represents the week day.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMAR WDSEL LL_RTC_ALMA_EnableWeekday

LL_RTC_ALMA_DisableWeekday

Function name

__STATIC_INLINE void LL_RTC_ALMA_DisableWeekday (RTC_TypeDef * RTCx)

Function description

Disable AlarmA Week day selection (DU[3:0] represents the date)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMAR WDSEL LL_RTC_ALMA_DisableWeekday

LL_RTC_ALMA_SetDay

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetDay (RTC_TypeDef * RTCx, uint32_t Day)

Function description

Set ALARM A Day in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Day:** Value between Min_Data=0x01 and Max_Data=0x31

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Day from binary to BCD format

Reference Manual to LL API cross reference:

- ALRMAR DT LL_RTC_ALMA_SetDay
- ALRMAR DU LL_RTC_ALMA_SetDay

LL_RTC_ALMA_GetDay

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetDay (RTC_TypeDef * RTCx)

Function description

Get ALARM A Day in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x31

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Day from BCD to Binary format

Reference Manual to LL API cross reference:

- ALRMAR DT LL_RTC_ALMA_GetDay
- ALRMAR DU LL_RTC_ALMA_GetDay

LL_RTC_ALMA_SetWeekDay

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetWeekDay (RTC_TypeDef * RTCx, uint32_t WeekDay)

Function description

Set ALARM A Weekday.

Parameters

- **RTCx:** RTC Instance
- **WeekDay:** This parameter can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMAR DU LL_RTC_ALMA_SetWeekDay

LL_RTC_ALMA_GetWeekDay

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetWeekDay (RTC_TypeDef * RTCx)

Function description

Get ALARM A Weekday.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY

Reference Manual to LL API cross reference:

- ALRMAR DU LL_RTC_ALMA_GetWeekDay

LL_RTC_ALMA_SetTimeFormat

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetTimeFormat (RTC_TypeDef * RTCx, uint32_t TimeFormat)

Function description

Set Alarm A time format (AM/24-hour or PM notation)

Parameters

- **RTCx:** RTC Instance
- **TimeFormat:** This parameter can be one of the following values:
  - LL_RTC_ALMA_TIME_FORMAT_AM
  - LL_RTC_ALMA_TIME_FORMAT_PM

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMAR PM LL_RTC_ALMA_SetTimeFormat

LL_RTC_ALMA_GetTimeFormat

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetTimeFormat (RTC_TypeDef * RTCx)

Function description

Get Alarm A time format (AM or PM notation)

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_ALMA_TIME_FORMAT_AM
  - LL_RTC_ALMA_TIME_FORMAT_PM

Reference Manual to LL API cross reference:

- ALRMAR PM LL_RTC_ALMA_GetTimeFormat

LL_RTC_ALMA_SetHour

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetHour (RTC_TypeDef * RTCx, uint32_t Hours)

Function description

Set ALARM A Hours in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Hours:** Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Hours from binary to BCD format

Reference Manual to LL API cross reference:

- ALRMAR HT LL_RTC_ALMA_SetHour
- ALRMAR HU LL_RTC_ALMA_SetHour

LL_RTC_ALMA_GetHour

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetHour (RTC_TypeDef * RTCx)

Function description

Get ALARM A Hours in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Hours from BCD to Binary format

Reference Manual to LL API cross reference:

- ALRMAR HT LL_RTC_ALMA_GetHour
- ALRMAR HU LL_RTC_ALMA_GetHour

LL_RTC_ALMA_SetMinute

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetMinute (RTC_TypeDef * RTCx, uint32_t Minutes)

Function description

Set ALARM A Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Minutes:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Minutes from binary to BCD format

Reference Manual to LL API cross reference:

- ALRMAR MNT LL_RTC_ALMA_SetMinute
- ALRMAR MNU LL_RTC_ALMA_SetMinute

LL_RTC_ALMA_GetMinute

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetMinute (RTC_TypeDef * RTCx)

Function description

Get ALARM A Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Minutes from BCD to Binary format

Reference Manual to LL API cross reference:

- ALRMAR MNT LL_RTC_ALMA_GetMinute
- ALRMAR MNU LL_RTC_ALMA_GetMinute

LL_RTC_ALMA_SetSecond

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetSecond (RTC_TypeDef * RTCx, uint32_t Seconds)

Function description

Set ALARM A Seconds in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Seconds:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Seconds from binary to BCD format

Reference Manual to LL API cross reference:

- ALRMAR ST LL_RTC_ALMA_SetSecond
- ALRMAR SU LL_RTC_ALMA_SetSecond

LL_RTC_ALMA_GetSecond

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetSecond (RTC_TypeDef * RTCx)

Function description

Get ALARM A Seconds in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Seconds from BCD to Binary format

Reference Manual to LL API cross reference:

- ALRMAR ST LL_RTC_ALMA_GetSecond
- ALRMAR SU LL_RTC_ALMA_GetSecond

LL_RTC_ALMA_ConfigTime

Function name

__STATIC_INLINE void LL_RTC_ALMA_ConfigTime (RTC_TypeDef * RTCx, uint32_t Format12_24, uint32_t Hours, uint32_t Minutes, uint32_t Seconds)

Function description

Set Alarm A Time (hour, minute and second) in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Format12_24:** This parameter can be one of the following values:
  - LL_RTC_ALMA_TIME_FORMAT_AM
  - LL_RTC_ALMA_TIME_FORMAT_PM
- **Hours:** Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23
- **Minutes:** Value between Min_Data=0x00 and Max_Data=0x59
- **Seconds:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMAR PM LL_RTC_ALMA_ConfigTime
- ALRMAR HT LL_RTC_ALMA_ConfigTime
- ALRMAR HU LL_RTC_ALMA_ConfigTime
- ALRMAR MNT LL_RTC_ALMA_ConfigTime
- ALRMAR MNU LL_RTC_ALMA_ConfigTime
- ALRMAR ST LL_RTC_ALMA_ConfigTime
- ALRMAR SU LL_RTC_ALMA_ConfigTime

LL_RTC_ALMA_GetTime

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetTime (RTC_TypeDef * RTCx)

Function description

Get Alarm B Time (hour, minute and second) in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Combination:** of hours, minutes and seconds.

Notes

- helper macros __LL_RTC_GET_HOUR, __LL_RTC_GET_MINUTE and __LL_RTC_GET_SECOND are available to get independently each parameter.

Reference Manual to LL API cross reference:

- ALRMAR HT LL_RTC_ALMA_GetTime
- ALRMAR HU LL_RTC_ALMA_GetTime
- ALRMAR MNT LL_RTC_ALMA_GetTime
- ALRMAR MNU LL_RTC_ALMA_GetTime
- ALRMAR ST LL_RTC_ALMA_GetTime
- ALRMAR SU LL_RTC_ALMA_GetTime

LL_RTC_ALMA_SetSubSecondMask

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetSubSecondMask (RTC_TypeDef * RTCx, uint32_t Mask)

Function description

Set Alarm A Mask the most-significant bits starting at this bit.

Parameters

- **RTCx:** RTC Instance
- **Mask:** Value between Min_Data=0x00 and Max_Data=0xF

Return values

- **None:**

Notes

- This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

Reference Manual to LL API cross reference:

- ALRMASSR MASKSS LL_RTC_ALMA_SetSubSecondMask

LL_RTC_ALMA_GetSubSecondMask

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetSubSecondMask (RTC_TypeDef * RTCx)

Function description

Get Alarm A Mask the most-significant bits starting at this bit.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xF

Reference Manual to LL API cross reference:

- ALRMASSR MASKSS LL_RTC_ALMA_GetSubSecondMask

LL_RTC_ALMA_SetSubSecond

Function name

__STATIC_INLINE void LL_RTC_ALMA_SetSubSecond (RTC_TypeDef * RTCx, uint32_t Subsecond)

Function description

Set Alarm A Sub seconds value.

Parameters

- **RTCx:** RTC Instance
- **Subsecond:** Value between Min_Data=0x00 and Max_Data=0x7FFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMASSR SS LL_RTC_ALMA_SetSubSecond

LL_RTC_ALMA_GetSubSecond

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMA_GetSubSecond (RTC_TypeDef * RTCx)

Function description

Get Alarm A Sub seconds value.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x7FFF

Reference Manual to LL API cross reference:

- ALRMASSR SS LL_RTC_ALMA_GetSubSecond

LL_RTC_ALMB_Enable

Function name

__STATIC_INLINE void LL_RTC_ALMB_Enable (RTC_TypeDef * RTCx)

Function description

Enable Alarm B.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ALRBE LL_RTC_ALMB_Enable

LL_RTC_ALMB_Disable

Function name

__STATIC_INLINE void LL_RTC_ALMB_Disable (RTC_TypeDef * RTCx)

Function description

Disable Alarm B.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ALRBE LL_RTC_ALMB_Disable

LL_RTC_ALMB_SetMask

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetMask (RTC_TypeDef * RTCx, uint32_t Mask)

Function description

Specify the Alarm B masks.

Parameters

- **RTCx:** RTC Instance
- **Mask:** This parameter can be a combination of the following values:
  - LL_RTC_ALMB_MASK_NONE
  - LL_RTC_ALMB_MASK_DATEWEEKDAY
  - LL_RTC_ALMB_MASK_HOURS
  - LL_RTC_ALMB_MASK_MINUTES
  - LL_RTC_ALMB_MASK_SECONDS
  - LL_RTC_ALMB_MASK_ALL

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMBR MSK4 LL_RTC_ALMB_SetMask
- ALRMBR MSK3 LL_RTC_ALMB_SetMask
- ALRMBR MSK2 LL_RTC_ALMB_SetMask
- ALRMBR MSK1 LL_RTC_ALMB_SetMask

LL_RTC_ALMB_GetMask

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetMask (RTC_TypeDef * RTCx)

Function description

Get the Alarm B masks.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be can be a combination of the following values:
  - LL_RTC_ALMB_MASK_NONE
  - LL_RTC_ALMB_MASK_DATEWEEKDAY
  - LL_RTC_ALMB_MASK_HOURS
  - LL_RTC_ALMB_MASK_MINUTES
  - LL_RTC_ALMB_MASK_SECONDS
  - LL_RTC_ALMB_MASK_ALL

Reference Manual to LL API cross reference:

- ALRMBR MSK4 LL_RTC_ALMB_GetMask
- ALRMBR MSK3 LL_RTC_ALMB_GetMask
- ALRMBR MSK2 LL_RTC_ALMB_GetMask
- ALRMBR MSK1 LL_RTC_ALMB_GetMask

LL_RTC_ALMB_EnableWeekday

Function name

__STATIC_INLINE void LL_RTC_ALMB_EnableWeekday (RTC_TypeDef * RTCx)

Function description

Enable AlarmB Week day selection (DU[3:0] represents the week day.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMBR WDSEL LL_RTC_ALMB_EnableWeekday

LL_RTC_ALMB_DisableWeekday

Function name

__STATIC_INLINE void LL_RTC_ALMB_DisableWeekday (RTC_TypeDef * RTCx)

Function description

Disable AlarmB Week day selection (DU[3:0] represents the date)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMBR WDSEL LL_RTC_ALMB_DisableWeekday

LL_RTC_ALMB_SetDay

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetDay (RTC_TypeDef * RTCx, uint32_t Day)

Function description

Set ALARM B Day in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Day:** Value between Min_Data=0x01 and Max_Data=0x31

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Day from binary to BCD format

Reference Manual to LL API cross reference:

- ALRMBR DT LL_RTC_ALMB_SetDay
- ALRMBR DU LL_RTC_ALMB_SetDay

LL_RTC_ALMB_GetDay

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetDay (RTC_TypeDef * RTCx)

Function description

Get ALARM B Day in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x31

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Day from BCD to Binary format

Reference Manual to LL API cross reference:

- ALRMBR DT LL_RTC_ALMB_GetDay
- ALRMBR DU LL_RTC_ALMB_GetDay

LL_RTC_ALMB_SetWeekDay

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetWeekDay (RTC_TypeDef * RTCx, uint32_t WeekDay)

Function description

Set ALARM B Weekday.

Parameters

- **RTCx:** RTC Instance
- **WeekDay:** This parameter can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMBR DU LL_RTC_ALMB_SetWeekDay

LL_RTC_ALMB_GetWeekDay

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetWeekDay (RTC_TypeDef * RTCx)

Function description

Get ALARM B Weekday.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY

Reference Manual to LL API cross reference:

- ALRMBR DU LL_RTC_ALMB_GetWeekDay

LL_RTC_ALMB_SetTimeFormat

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetTimeFormat (RTC_TypeDef * RTCx, uint32_t TimeFormat)

Function description

Set ALARM B time format (AM/24-hour or PM notation)

Parameters

- **RTCx:** RTC Instance
- **TimeFormat:** This parameter can be one of the following values:
  - LL_RTC_ALMB_TIME_FORMAT_AM
  - LL_RTC_ALMB_TIME_FORMAT_PM

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMBR PM LL_RTC_ALMB_SetTimeFormat

LL_RTC_ALMB_GetTimeFormat

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetTimeFormat (RTC_TypeDef * RTCx)

Function description

Get ALARM B time format (AM or PM notation)

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_ALMB_TIME_FORMAT_AM
  - LL_RTC_ALMB_TIME_FORMAT_PM

Reference Manual to LL API cross reference:

- ALRMBR PM LL_RTC_ALMB_GetTimeFormat

LL_RTC_ALMB_SetHour

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetHour (RTC_TypeDef * RTCx, uint32_t Hours)

Function description

Set ALARM B Hours in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Hours:** Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Hours from binary to BCD format

Reference Manual to LL API cross reference:

- ALRMBR HT LL_RTC_ALMB_SetHour
- ALRMBR HU LL_RTC_ALMB_SetHour

LL_RTC_ALMB_GetHour

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetHour (RTC_TypeDef * RTCx)

Function description

Get ALARM B Hours in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Hours from BCD to Binary format

Reference Manual to LL API cross reference:

- ALRMBR HT LL_RTC_ALMB_GetHour
- ALRMBR HU LL_RTC_ALMB_GetHour

LL_RTC_ALMB_SetMinute

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetMinute (RTC_TypeDef * RTCx, uint32_t Minutes)

Function description

Set ALARM B Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Minutes:** between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Minutes from binary to BCD format

Reference Manual to LL API cross reference:

- ALRMBR MNT LL_RTC_ALMB_SetMinute
- ALRMBR MNU LL_RTC_ALMB_SetMinute

LL_RTC_ALMB_GetMinute

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetMinute (RTC_TypeDef * RTCx)

Function description

Get ALARM B Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Minutes from BCD to Binary format

Reference Manual to LL API cross reference:

- ALRMBR MNT LL_RTC_ALMB_GetMinute
- ALRMBR MNU LL_RTC_ALMB_GetMinute

LL_RTC_ALMB_SetSecond

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetSecond (RTC_TypeDef * RTCx, uint32_t Seconds)

Function description

Set ALARM B Seconds in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Seconds:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Notes

- helper macro __LL_RTC_CONVERT_BIN2BCD is available to convert Seconds from binary to BCD format

Reference Manual to LL API cross reference:

- ALRMBR ST LL_RTC_ALMB_SetSecond
- ALRMBR SU LL_RTC_ALMB_SetSecond

LL_RTC_ALMB_GetSecond

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetSecond (RTC_TypeDef * RTCx)

Function description

Get ALARM B Seconds in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Seconds from BCD to Binary format

Reference Manual to LL API cross reference:

- ALRMBR ST LL_RTC_ALMB_GetSecond
- ALRMBR SU LL_RTC_ALMB_GetSecond

LL_RTC_ALMB_ConfigTime

Function name

__STATIC_INLINE void LL_RTC_ALMB_ConfigTime (RTC_TypeDef * RTCx, uint32_t Format12_24, uint32_t Hours, uint32_t Minutes, uint32_t Seconds)

Function description

Set Alarm B Time (hour, minute and second) in BCD format.

Parameters

- **RTCx:** RTC Instance
- **Format12_24:** This parameter can be one of the following values:
  - LL_RTC_ALMB_TIME_FORMAT_AM
  - LL_RTC_ALMB_TIME_FORMAT_PM
- **Hours:** Value between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23
- **Minutes:** Value between Min_Data=0x00 and Max_Data=0x59
- **Seconds:** Value between Min_Data=0x00 and Max_Data=0x59

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMBR PM LL_RTC_ALMB_ConfigTime
- ALRMBR HT LL_RTC_ALMB_ConfigTime
- ALRMBR HU LL_RTC_ALMB_ConfigTime
- ALRMBR MNT LL_RTC_ALMB_ConfigTime
- ALRMBR MNU LL_RTC_ALMB_ConfigTime
- ALRMBR ST LL_RTC_ALMB_ConfigTime
- ALRMBR SU LL_RTC_ALMB_ConfigTime

LL_RTC_ALMB_GetTime

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetTime (RTC_TypeDef * RTCx)

Function description

Get Alarm B Time (hour, minute and second) in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Combination:** of hours, minutes and seconds.

Notes

- helper macros __LL_RTC_GET_HOUR, __LL_RTC_GET_MINUTE and __LL_RTC_GET_SECOND are available to get independently each parameter.

Reference Manual to LL API cross reference:

- ALRMBR HT LL_RTC_ALMB_GetTime
- ALRMBR HU LL_RTC_ALMB_GetTime
- ALRMBR MNT LL_RTC_ALMB_GetTime
- ALRMBR MNU LL_RTC_ALMB_GetTime
- ALRMBR ST LL_RTC_ALMB_GetTime
- ALRMBR SU LL_RTC_ALMB_GetTime

LL_RTC_ALMB_SetSubSecondMask

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetSubSecondMask (RTC_TypeDef * RTCx, uint32_t Mask)

Function description

Set Alarm B Mask the most-significant bits starting at this bit.

Parameters

- **RTCx:** RTC Instance
- **Mask:** Value between Min_Data=0x00 and Max_Data=0xF

Return values

- **None:**

Notes

- This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

Reference Manual to LL API cross reference:

- ALRMBSSR MASKSS LL_RTC_ALMB_SetSubSecondMask

LL_RTC_ALMB_GetSubSecondMask

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetSubSecondMask (RTC_TypeDef * RTCx)

Function description

Get Alarm B Mask the most-significant bits starting at this bit.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xF

Reference Manual to LL API cross reference:

- ALRMBSSR MASKSS LL_RTC_ALMB_GetSubSecondMask

LL_RTC_ALMB_SetSubSecond

Function name

__STATIC_INLINE void LL_RTC_ALMB_SetSubSecond (RTC_TypeDef * RTCx, uint32_t Subsecond)

Function description

Set Alarm B Sub seconds value.

Parameters

- **RTCx:** RTC Instance
- **Subsecond:** Value between Min_Data=0x00 and Max_Data=0x7FFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- ALRMBSSR SS LL_RTC_ALMB_SetSubSecond

LL_RTC_ALMB_GetSubSecond

Function name

__STATIC_INLINE uint32_t LL_RTC_ALMB_GetSubSecond (RTC_TypeDef * RTCx)

Function description

Get Alarm B Sub seconds value.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x7FFF

Reference Manual to LL API cross reference:

- ALRMBSSR SS LL_RTC_ALMB_GetSubSecond

LL_RTC_TS_Enable

Function name

__STATIC_INLINE void LL_RTC_TS_Enable (RTC_TypeDef * RTCx)

Function description

Enable Timestamp.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR TSE LL_RTC_TS_Enable

LL_RTC_TS_Disable

Function name

__STATIC_INLINE void LL_RTC_TS_Disable (RTC_TypeDef * RTCx)

Function description

Disable Timestamp.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR TSE LL_RTC_TS_Disable

LL_RTC_TS_SetActiveEdge

Function name

__STATIC_INLINE void LL_RTC_TS_SetActiveEdge (RTC_TypeDef * RTCx, uint32_t Edge)

Function description

Set Time-stamp event active edge.

Parameters

- **RTCx:** RTC Instance
- **Edge:** This parameter can be one of the following values:
  - LL_RTC_TIMESTAMP_EDGE_RISING
  - LL_RTC_TIMESTAMP_EDGE_FALLING

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting

Reference Manual to LL API cross reference:

- CR TSEDGE LL_RTC_TS_SetActiveEdge

LL_RTC_TS_GetActiveEdge

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetActiveEdge (RTC_TypeDef * RTCx)

Function description

Get Time-stamp event active edge.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_TIMESTAMP_EDGE_RISING
  - LL_RTC_TIMESTAMP_EDGE_FALLING

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR TSEDGE LL_RTC_TS_GetActiveEdge

LL_RTC_TS_GetTimeFormat

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetTimeFormat (RTC_TypeDef * RTCx)

Function description

Get Timestamp AM/PM notation (AM or 24-hour format)

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_TS_TIME_FORMAT_AM
  - LL_RTC_TS_TIME_FORMAT_PM

Reference Manual to LL API cross reference:

- TSTR PM LL_RTC_TS_GetTimeFormat

LL_RTC_TS_GetHour

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetHour (RTC_TypeDef * RTCx)

Function description

Get Timestamp Hours in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x12 or between Min_Data=0x00 and Max_Data=0x23

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Hours from BCD to Binary format

Reference Manual to LL API cross reference:

- TSTR HT LL_RTC_TS_GetHour
- TSTR HU LL_RTC_TS_GetHour

LL_RTC_TS_GetMinute

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetMinute (RTC_TypeDef * RTCx)

Function description

Get Timestamp Minutes in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Minutes from BCD to Binary format

Reference Manual to LL API cross reference:

- TSTR MNT LL_RTC_TS_GetMinute
- TSTR MNU LL_RTC_TS_GetMinute

LL_RTC_TS_GetSecond

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetSecond (RTC_TypeDef * RTCx)

Function description

Get Timestamp Seconds in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x59

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Seconds from BCD to Binary format

Reference Manual to LL API cross reference:

- TSTR ST LL_RTC_TS_GetSecond
- TSTR SU LL_RTC_TS_GetSecond

LL_RTC_TS_GetTime

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetTime (RTC_TypeDef * RTCx)

Function description

Get Timestamp time (hour, minute and second) in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Combination:** of hours, minutes and seconds.

Notes

- helper macros __LL_RTC_GET_HOUR, __LL_RTC_GET_MINUTE and __LL_RTC_GET_SECOND are available to get independently each parameter.

Reference Manual to LL API cross reference:

- TSTR HT LL_RTC_TS_GetTime
- TSTR HU LL_RTC_TS_GetTime
- TSTR MNT LL_RTC_TS_GetTime
- TSTR MNU LL_RTC_TS_GetTime
- TSTR ST LL_RTC_TS_GetTime
- TSTR SU LL_RTC_TS_GetTime

LL_RTC_TS_GetWeekDay

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetWeekDay (RTC_TypeDef * RTCx)

Function description

Get Timestamp Week day.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY

Reference Manual to LL API cross reference:

- TSDR WDU LL_RTC_TS_GetWeekDay

LL_RTC_TS_GetMonth

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetMonth (RTC_TypeDef * RTCx)

Function description

Get Timestamp Month in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_MONTH_JANUARY
  - LL_RTC_MONTH_FEBRUARY
  - LL_RTC_MONTH_MARCH
  - LL_RTC_MONTH_APRIL
  - LL_RTC_MONTH_MAY
  - LL_RTC_MONTH_JUNE
  - LL_RTC_MONTH_JULY
  - LL_RTC_MONTH_AUGUST
  - LL_RTC_MONTH_SEPTEMBER
  - LL_RTC_MONTH_OCTOBER
  - LL_RTC_MONTH_NOVEMBER
  - LL_RTC_MONTH_DECEMBER

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Month from BCD to Binary format

Reference Manual to LL API cross reference:

- TSDR MT LL_RTC_TS_GetMonth
- TSDR MU LL_RTC_TS_GetMonth

LL_RTC_TS_GetDay

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetDay (RTC_TypeDef * RTCx)

Function description

Get Timestamp Day in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x01 and Max_Data=0x31

Notes

- helper macro __LL_RTC_CONVERT_BCD2BIN is available to convert Day from BCD to Binary format

Reference Manual to LL API cross reference:

- TSDR DT LL_RTC_TS_GetDay
- TSDR DU LL_RTC_TS_GetDay

LL_RTC_TS_GetDate

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetDate (RTC_TypeDef * RTCx)

Function description

Get Timestamp date (WeekDay, Day and Month) in BCD format.

Parameters

- **RTCx:** RTC Instance

Return values

- **Combination:** of Weekday, Day and Month

Notes

- helper macros __LL_RTC_GET_WEEKDAY, __LL_RTC_GET_MONTH, and __LL_RTC_GET_DAY are available to get independently each parameter.

Reference Manual to LL API cross reference:

- TSDR WDU LL_RTC_TS_GetDate
- TSDR MT LL_RTC_TS_GetDate
- TSDR MU LL_RTC_TS_GetDate
- TSDR DT LL_RTC_TS_GetDate
- TSDR DU LL_RTC_TS_GetDate

LL_RTC_TS_GetSubSecond

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetSubSecond (RTC_TypeDef * RTCx)

Function description

Get time-stamp sub second value.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- TSSSR SS LL_RTC_TS_GetSubSecond

LL_RTC_TS_EnableOnTamper

Function name

__STATIC_INLINE void LL_RTC_TS_EnableOnTamper (RTC_TypeDef * RTCx)

Function description

Activate timestamp on tamper detection event.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPTS LL_RTC_TS_EnableOnTamper

LL_RTC_TS_DisableOnTamper

Function name

__STATIC_INLINE void LL_RTC_TS_DisableOnTamper (RTC_TypeDef * RTCx)

Function description

Disable timestamp on tamper detection event.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPTS LL_RTC_TS_DisableOnTamper

LL_RTC_TS_SetPin

Function name

__STATIC_INLINE void LL_RTC_TS_SetPin (RTC_TypeDef * RTCx, uint32_t TSPin)

Function description

Set timestamp Pin.

Parameters

- **RTCx:** RTC Instance
- **TSPin:** specifies the RTC TimeStamp Pin. This parameter can be one of the following values:
  - LL_RTC_TimeStampPin_Default: RTC_AF1 is used as RTC TimeStamp.
  - LL_RTC_TimeStampPin_Pos1: RTC_AF2 is selected as RTC TimeStamp. (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TSINSEL LL_RTC_TS_SetPin

LL_RTC_TS_GetPin

Function name

__STATIC_INLINE uint32_t LL_RTC_TS_GetPin (RTC_TypeDef * RTCx)

Function description

Get timestamp Pin.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_TimeStampPin_Default: RTC_AF1 is used as RTC TimeStamp Pin.
  - LL_RTC_TimeStampPin_Pos1: RTC_AF2 is selected as RTC TimeStamp Pin. (*)

  (*) value not defined in all devices.
- **None:**

Reference Manual to LL API cross reference:

- TAFCR TSINSEL LL_RTC_TS_GetPin

LL_RTC_TAMPER_Enable

Function name

__STATIC_INLINE void LL_RTC_TAMPER_Enable (RTC_TypeDef * RTCx, uint32_t Tamper)

Function description

Enable RTC_TAMPx input detection.

Parameters

- **RTCx:** RTC Instance
- **Tamper:** This parameter can be a combination of the following values:
  - LL_RTC_TAMPER_1
  - LL_RTC_TAMPER_2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMP1E LL_RTC_TAMPER_Enable
- TAFCR TAMP2E LL_RTC_TAMPER_Enable
-

LL_RTC_TAMPER_Disable

Function name

__STATIC_INLINE void LL_RTC_TAMPER_Disable (RTC_TypeDef * RTCx, uint32_t Tamper)

Function description

Clear RTC_TAMPx input detection.

Parameters

- **RTCx:** RTC Instance
- **Tamper:** This parameter can be a combination of the following values:
  - LL_RTC_TAMPER_1
  - LL_RTC_TAMPER_2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMP1E LL_RTC_TAMPER_Disable
- TAFCR TAMP2E LL_RTC_TAMPER_Disable
-

LL_RTC_TAMPER_DisablePullUp

Function name

__STATIC_INLINE void LL_RTC_TAMPER_DisablePullUp (RTC_TypeDef * RTCx)

Function description

Disable RTC_TAMPx pull-up disable (Disable precharge of RTC_TAMPx pins)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPPUDIS LL_RTC_TAMPER_DisablePullUp

LL_RTC_TAMPER_EnablePullUp

Function name

__STATIC_INLINE void LL_RTC_TAMPER_EnablePullUp (RTC_TypeDef * RTCx)

Function description

Enable RTC_TAMPx pull-up disable (Precharge RTC_TAMPx pins before sampling)

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPPUDIS LL_RTC_TAMPER_EnablePullUp

LL_RTC_TAMPER_SetPrecharge

Function name

__STATIC_INLINE void LL_RTC_TAMPER_SetPrecharge (RTC_TypeDef * RTCx, uint32_t Duration)

Function description

Set RTC_TAMPx precharge duration.

Parameters

- **RTCx:** RTC Instance
- **Duration:** This parameter can be one of the following values:
  - LL_RTC_TAMPER_DURATION_1RTCCLK
  - LL_RTC_TAMPER_DURATION_2RTCCLK
  - LL_RTC_TAMPER_DURATION_4RTCCLK
  - LL_RTC_TAMPER_DURATION_8RTCCLK

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPPRCH LL_RTC_TAMPER_SetPrecharge

LL_RTC_TAMPER_GetPrecharge

Function name

__STATIC_INLINE uint32_t LL_RTC_TAMPER_GetPrecharge (RTC_TypeDef * RTCx)

Function description

Get RTC_TAMPx precharge duration.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_TAMPER_DURATION_1RTCCLK
  - LL_RTC_TAMPER_DURATION_2RTCCLK
  - LL_RTC_TAMPER_DURATION_4RTCCLK
  - LL_RTC_TAMPER_DURATION_8RTCCLK

Reference Manual to LL API cross reference:

- TAFCR TAMPPRCH LL_RTC_TAMPER_GetPrecharge

LL_RTC_TAMPER_SetFilterCount

Function name

__STATIC_INLINE void LL_RTC_TAMPER_SetFilterCount (RTC_TypeDef * RTCx, uint32_t FilterCount)

Function description

Set RTC_TAMPx filter count.

Parameters

- **RTCx:** RTC Instance
- **FilterCount:** This parameter can be one of the following values:
  - LL_RTC_TAMPER_FILTER_DISABLE
  - LL_RTC_TAMPER_FILTER_2SAMPLE
  - LL_RTC_TAMPER_FILTER_4SAMPLE
  - LL_RTC_TAMPER_FILTER_8SAMPLE

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPFLT LL_RTC_TAMPER_SetFilterCount

LL_RTC_TAMPER_GetFilterCount

Function name

__STATIC_INLINE uint32_t LL_RTC_TAMPER_GetFilterCount (RTC_TypeDef * RTCx)

Function description

Get RTC_TAMPx filter count.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_TAMPER_FILTER_DISABLE
  - LL_RTC_TAMPER_FILTER_2SAMPLE
  - LL_RTC_TAMPER_FILTER_4SAMPLE
  - LL_RTC_TAMPER_FILTER_8SAMPLE

Reference Manual to LL API cross reference:

- TAFCR TAMPFLT LL_RTC_TAMPER_GetFilterCount

LL_RTC_TAMPER_SetSamplingFreq

Function name

__STATIC_INLINE void LL_RTC_TAMPER_SetSamplingFreq (RTC_TypeDef * RTCx, uint32_t SamplingFreq)

Function description

Set Tamper sampling frequency.

Parameters

- **RTCx:** RTC Instance
- **SamplingFreq:** This parameter can be one of the following values:
  - LL_RTC_TAMPER_SAMPLFREQDIV_32768
  - LL_RTC_TAMPER_SAMPLFREQDIV_16384
  - LL_RTC_TAMPER_SAMPLFREQDIV_8192
  - LL_RTC_TAMPER_SAMPLFREQDIV_4096
  - LL_RTC_TAMPER_SAMPLFREQDIV_2048
  - LL_RTC_TAMPER_SAMPLFREQDIV_1024
  - LL_RTC_TAMPER_SAMPLFREQDIV_512
  - LL_RTC_TAMPER_SAMPLFREQDIV_256

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPFREQ LL_RTC_TAMPER_SetSamplingFreq

LL_RTC_TAMPER_GetSamplingFreq

Function name

__STATIC_INLINE uint32_t LL_RTC_TAMPER_GetSamplingFreq (RTC_TypeDef * RTCx)

Function description

Get Tamper sampling frequency.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_TAMPER_SAMPLFREQDIV_32768
  - LL_RTC_TAMPER_SAMPLFREQDIV_16384
  - LL_RTC_TAMPER_SAMPLFREQDIV_8192
  - LL_RTC_TAMPER_SAMPLFREQDIV_4096
  - LL_RTC_TAMPER_SAMPLFREQDIV_2048
  - LL_RTC_TAMPER_SAMPLFREQDIV_1024
  - LL_RTC_TAMPER_SAMPLFREQDIV_512
  - LL_RTC_TAMPER_SAMPLFREQDIV_256

Reference Manual to LL API cross reference:

- TAFCR TAMPFREQ LL_RTC_TAMPER_GetSamplingFreq

LL_RTC_TAMPER_EnableActiveLevel

Function name

__STATIC_INLINE void LL_RTC_TAMPER_EnableActiveLevel (RTC_TypeDef * RTCx, uint32_t Tamper)

Function description

Enable Active level for Tamper input.

Parameters

- **RTCx:** RTC Instance
- **Tamper:** This parameter can be a combination of the following values:
  - LL_RTC_TAMPER_ACTIVELEVEL_TAMP1
  - LL_RTC_TAMPER_ACTIVELEVEL_TAMP2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMP1TRG LL_RTC_TAMPER_EnableActiveLevel
- TAFCR TAMP2TRG LL_RTC_TAMPER_EnableActiveLevel
-

LL_RTC_TAMPER_DisableActiveLevel

Function name

__STATIC_INLINE void LL_RTC_TAMPER_DisableActiveLevel (RTC_TypeDef * RTCx, uint32_t Tamper)

Function description

Disable Active level for Tamper input.

Parameters

- **RTCx:** RTC Instance
- **Tamper:** This parameter can be a combination of the following values:
  - LL_RTC_TAMPER_ACTIVELEVEL_TAMP1
  - LL_RTC_TAMPER_ACTIVELEVEL_TAMP2 (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMP1TRG LL_RTC_TAMPER_DisableActiveLevel
- TAFCR TAMP2TRG LL_RTC_TAMPER_DisableActiveLevel
-

LL_RTC_TAMPER_SetPin

Function name

__STATIC_INLINE void LL_RTC_TAMPER_SetPin (RTC_TypeDef * RTCx, uint32_t TamperPin)

Function description

Set Tamper Pin.

Parameters

- **RTCx:** RTC Instance
- **TamperPin:** specifies the RTC Tamper Pin. This parameter can be one of the following values:
  - LL_RTC_TamperPin_Default: RTC_AF1 is used as RTC Tamper.
  - LL_RTC_TamperPin_Pos1: RTC_AF2 is selected as RTC Tamper. (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMP1INSEL LL_RTC_TAMPER_SetPin

LL_RTC_TAMPER_GetPin

Function name

__STATIC_INLINE uint32_t LL_RTC_TAMPER_GetPin (RTC_TypeDef * RTCx)

Function description

Get Tamper Pin.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_TamperPin_Default: RTC_AF1 is used as RTC Tamper Pin.
  - LL_RTC_TamperPin_Pos1: RTC_AF2 is selected as RTC Tamper Pin. (*)

  (*) value not defined in all devices.
- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMP1INSEL LL_RTC_TAMPER_GetPin

LL_RTC_WAKEUP_Enable

Function name

__STATIC_INLINE void LL_RTC_WAKEUP_Enable (RTC_TypeDef * RTCx)

Function description

Enable Wakeup timer.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR WUTE LL_RTC_WAKEUP_Enable

LL_RTC_WAKEUP_Disable

Function name

__STATIC_INLINE void LL_RTC_WAKEUP_Disable (RTC_TypeDef * RTCx)

Function description

Disable Wakeup timer.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR WUTE LL_RTC_WAKEUP_Disable

LL_RTC_WAKEUP_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_RTC_WAKEUP_IsEnabled (RTC_TypeDef * RTCx)

Function description

Check if Wakeup timer is enabled or not.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR WUTE LL_RTC_WAKEUP_IsEnabled

LL_RTC_WAKEUP_SetClock

Function name

__STATIC_INLINE void LL_RTC_WAKEUP_SetClock (RTC_TypeDef * RTCx, uint32_t WakeupClock)

Function description

Select Wakeup clock.

Parameters

- **RTCx:** RTC Instance
- **WakeupClock:** This parameter can be one of the following values:
  - LL_RTC_WAKEUPCLOCK_DIV_16
  - LL_RTC_WAKEUPCLOCK_DIV_8
  - LL_RTC_WAKEUPCLOCK_DIV_4
  - LL_RTC_WAKEUPCLOCK_DIV_2
  - LL_RTC_WAKEUPCLOCK_CKSPRE
  - LL_RTC_WAKEUPCLOCK_CKSPRE_WUT

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- Bit can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1

Reference Manual to LL API cross reference:

- CR WUCKSEL LL_RTC_WAKEUP_SetClock

LL_RTC_WAKEUP_GetClock

Function name

__STATIC_INLINE uint32_t LL_RTC_WAKEUP_GetClock (RTC_TypeDef * RTCx)

Function description

Get Wakeup clock.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_WAKEUPCLOCK_DIV_16
  - LL_RTC_WAKEUPCLOCK_DIV_8
  - LL_RTC_WAKEUPCLOCK_DIV_4
  - LL_RTC_WAKEUPCLOCK_DIV_2
  - LL_RTC_WAKEUPCLOCK_CKSPRE
  - LL_RTC_WAKEUPCLOCK_CKSPRE_WUT

Reference Manual to LL API cross reference:

- CR WUCKSEL LL_RTC_WAKEUP_GetClock

LL_RTC_WAKEUP_SetAutoReload

Function name

__STATIC_INLINE void LL_RTC_WAKEUP_SetAutoReload (RTC_TypeDef * RTCx, uint32_t Value)

Function description

Set Wakeup auto-reload value.

Parameters

- **RTCx:** RTC Instance
- **Value:** Value between Min_Data=0x00 and Max_Data=0xFFFF

Return values

- **None:**

Notes

- Bit can be written only when WUTWF is set to 1 in RTC_ISR

Reference Manual to LL API cross reference:

- WUTR WUT LL_RTC_WAKEUP_SetAutoReload

LL_RTC_WAKEUP_GetAutoReload

Function name

__STATIC_INLINE uint32_t LL_RTC_WAKEUP_GetAutoReload (RTC_TypeDef * RTCx)

Function description

Get Wakeup auto-reload value.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- WUTR WUT LL_RTC_WAKEUP_GetAutoReload

LL_RTC_BAK_SetRegister

Function name

__STATIC_INLINE void LL_RTC_BAK_SetRegister (RTC_TypeDef * RTCx, uint32_t BackupRegister, uint32_t Data)

Function description

Writes a data in a specified RTC Backup data register.

Parameters

- **RTCx:** RTC Instance
- **BackupRegister:** This parameter can be one of the following values:
  - LL_RTC_BKP_DR0
  - LL_RTC_BKP_DR1
  - LL_RTC_BKP_DR2
  - LL_RTC_BKP_DR3
  - LL_RTC_BKP_DR4
  - LL_RTC_BKP_DR5
  - LL_RTC_BKP_DR6
  - LL_RTC_BKP_DR7
  - LL_RTC_BKP_DR8
  - LL_RTC_BKP_DR9
  - LL_RTC_BKP_DR10
  - LL_RTC_BKP_DR11
  - LL_RTC_BKP_DR12
  - LL_RTC_BKP_DR13
  - LL_RTC_BKP_DR14
  - LL_RTC_BKP_DR15
  - LL_RTC_BKP_DR16
  - LL_RTC_BKP_DR17
  - LL_RTC_BKP_DR18
  - LL_RTC_BKP_DR19
- **Data:** Value between Min_Data=0x00 and Max_Data=0xFFFFFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- BKPxR BKP LL_RTC_BAK_SetRegister

LL_RTC_BAK_GetRegister

Function name

__STATIC_INLINE uint32_t LL_RTC_BAK_GetRegister (RTC_TypeDef * RTCx, uint32_t BackupRegister)

Function description

Reads data from the specified RTC Backup data Register.

Parameters

- **RTCx:** RTC Instance
- **BackupRegister:** This parameter can be one of the following values:
  - LL_RTC_BKP_DR0
  - LL_RTC_BKP_DR1
  - LL_RTC_BKP_DR2
  - LL_RTC_BKP_DR3
  - LL_RTC_BKP_DR4
  - LL_RTC_BKP_DR5
  - LL_RTC_BKP_DR6
  - LL_RTC_BKP_DR7
  - LL_RTC_BKP_DR8
  - LL_RTC_BKP_DR9
  - LL_RTC_BKP_DR10
  - LL_RTC_BKP_DR11
  - LL_RTC_BKP_DR12
  - LL_RTC_BKP_DR13
  - LL_RTC_BKP_DR14
  - LL_RTC_BKP_DR15
  - LL_RTC_BKP_DR16
  - LL_RTC_BKP_DR17
  - LL_RTC_BKP_DR18
  - LL_RTC_BKP_DR19

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFFFFFFFF

Reference Manual to LL API cross reference:

- BKPxR BKP LL_RTC_BAK_GetRegister

LL_RTC_CAL_SetOutputFreq

Function name

__STATIC_INLINE void LL_RTC_CAL_SetOutputFreq (RTC_TypeDef * RTCx, uint32_t Frequency)

Function description

Set Calibration output frequency (1 Hz or 512 Hz)

Parameters

- **RTCx:** RTC Instance
- **Frequency:** This parameter can be one of the following values:
  - LL_RTC_CALIB_OUTPUT_NONE
  - LL_RTC_CALIB_OUTPUT_1HZ
  - LL_RTC_CALIB_OUTPUT_512HZ

Return values

- **None:**

Notes

- Bits are write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR COE LL_RTC_CAL_SetOutputFreq
- CR COSEL LL_RTC_CAL_SetOutputFreq

LL_RTC_CAL_GetOutputFreq

Function name

__STATIC_INLINE uint32_t LL_RTC_CAL_GetOutputFreq (RTC_TypeDef * RTCx)

Function description

Get Calibration output frequency (1 Hz or 512 Hz)

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_CALIB_OUTPUT_NONE
  - LL_RTC_CALIB_OUTPUT_1HZ
  - LL_RTC_CALIB_OUTPUT_512HZ

Reference Manual to LL API cross reference:

- CR COE LL_RTC_CAL_GetOutputFreq
- CR COSEL LL_RTC_CAL_GetOutputFreq

LL_RTC_CAL_EnableCoarseDigital

Function name

__STATIC_INLINE void LL_RTC_CAL_EnableCoarseDigital (RTC_TypeDef * RTCx)

Function description

Enable Coarse digital calibration.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)

Reference Manual to LL API cross reference:

- CR DCE LL_RTC_CAL_EnableCoarseDigital

LL_RTC_CAL_DisableCoarseDigital

Function name

__STATIC_INLINE void LL_RTC_CAL_DisableCoarseDigital (RTC_TypeDef * RTCx)

Function description

Disable Coarse digital calibration.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)

Reference Manual to LL API cross reference:

- CR DCE LL_RTC_CAL_DisableCoarseDigital

LL_RTC_CAL_ConfigCoarseDigital

Function name

__STATIC_INLINE void LL_RTC_CAL_ConfigCoarseDigital (RTC_TypeDef * RTCx, uint32_t Sign, uint32_t Value)

Function description

Set the coarse digital calibration.

Parameters

- **RTCx:** RTC Instance
- **Sign:** This parameter can be one of the following values:
  - LL_RTC_CALIB_SIGN_POSITIVE
  - LL_RTC_CALIB_SIGN_NEGATIVE
- **Value:** value of coarse calibration expressed in ppm (coded on 5 bits)

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- It can be written in initialization mode only (LL_RTC_EnableInitMode function)
- This Calibration value should be between 0 and 63 when using negative sign with a 2-ppm step.
- This Calibration value should be between 0 and 126 when using positive sign with a 4-ppm step.

Reference Manual to LL API cross reference:

- CALIBR DCS LL_RTC_CAL_ConfigCoarseDigital
- CALIBR DC LL_RTC_CAL_ConfigCoarseDigital

LL_RTC_CAL_GetCoarseDigitalValue

Function name

__STATIC_INLINE uint32_t LL_RTC_CAL_GetCoarseDigitalValue (RTC_TypeDef * RTCx)

Function description

Get the coarse digital calibration value.

Parameters

- **RTCx:** RTC Instance

Return values

- **value:** of coarse calibration expressed in ppm (coded on 5 bits)

Reference Manual to LL API cross reference:

- CALIBR DC LL_RTC_CAL_GetCoarseDigitalValue

LL_RTC_CAL_GetCoarseDigitalSign

Function name

__STATIC_INLINE uint32_t LL_RTC_CAL_GetCoarseDigitalSign (RTC_TypeDef * RTCx)

Function description

Get the coarse digital calibration sign.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_CALIB_SIGN_POSITIVE
  - LL_RTC_CALIB_SIGN_NEGATIVE

Reference Manual to LL API cross reference:

- CALIBR DCS LL_RTC_CAL_GetCoarseDigitalSign

LL_RTC_CAL_SetPulse

Function name

__STATIC_INLINE void LL_RTC_CAL_SetPulse (RTC_TypeDef * RTCx, uint32_t Pulse)

Function description

Insert or not One RTCCLK pulse every 2exp11 pulses (frequency increased by 488.5 ppm)

Parameters

- **RTCx:** RTC Instance
- **Pulse:** This parameter can be one of the following values:
  - LL_RTC_CALIB_INSERTPULSE_NONE
  - LL_RTC_CALIB_INSERTPULSE_SET

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- Bit can be written only when RECALPF is set to 0 in RTC_ISR

Reference Manual to LL API cross reference:

- CALR CALP LL_RTC_CAL_SetPulse

LL_RTC_CAL_IsPulseInserted

Function name

__STATIC_INLINE uint32_t LL_RTC_CAL_IsPulseInserted (RTC_TypeDef * RTCx)

Function description

Check if one RTCCLK has been inserted or not every 2exp11 pulses (frequency increased by 488.5 ppm)

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CALR CALP LL_RTC_CAL_IsPulseInserted

LL_RTC_CAL_SetPeriod

Function name

__STATIC_INLINE void LL_RTC_CAL_SetPeriod (RTC_TypeDef * RTCx, uint32_t Period)

Function description

Set the calibration cycle period.

Parameters

- **RTCx:** RTC Instance
- **Period:** This parameter can be one of the following values:
  - LL_RTC_CALIB_PERIOD_32SEC
  - LL_RTC_CALIB_PERIOD_16SEC
  - LL_RTC_CALIB_PERIOD_8SEC

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- Bit can be written only when RECALPF is set to 0 in RTC_ISR

Reference Manual to LL API cross reference:

- CALR CALW8 LL_RTC_CAL_SetPeriod
- CALR CALW16 LL_RTC_CAL_SetPeriod

LL_RTC_CAL_GetPeriod

Function name

__STATIC_INLINE uint32_t LL_RTC_CAL_GetPeriod (RTC_TypeDef * RTCx)

Function description

Get the calibration cycle period.

Parameters

- **RTCx:** RTC Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_RTC_CALIB_PERIOD_32SEC
  - LL_RTC_CALIB_PERIOD_16SEC
  - LL_RTC_CALIB_PERIOD_8SEC

Reference Manual to LL API cross reference:

- CALR CALW8 LL_RTC_CAL_GetPeriod
- CALR CALW16 LL_RTC_CAL_GetPeriod

LL_RTC_CAL_SetMinus

Function name

__STATIC_INLINE void LL_RTC_CAL_SetMinus (RTC_TypeDef * RTCx, uint32_t CalibMinus)

Function description

Set Calibration minus.

Parameters

- **RTCx:** RTC Instance
- **CalibMinus:** Value between Min_Data=0x00 and Max_Data=0x1FF

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.
- Bit can be written only when RECALPF is set to 0 in RTC_ISR

Reference Manual to LL API cross reference:

- CALR CALM LL_RTC_CAL_SetMinus

LL_RTC_CAL_GetMinus

Function name

__STATIC_INLINE uint32_t LL_RTC_CAL_GetMinus (RTC_TypeDef * RTCx)

Function description

Get Calibration minus.

Parameters

- **RTCx:** RTC Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data= 0x1FF

Reference Manual to LL API cross reference:

- CALR CALM LL_RTC_CAL_GetMinus

LL_RTC_IsActiveFlag_RECALP

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_RECALP (RTC_TypeDef * RTCx)

Function description

Get Recalibration pending Flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR RECALPF LL_RTC_IsActiveFlag_RECALP

LL_RTC_IsActiveFlag_TAMP2

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TAMP2 (RTC_TypeDef * RTCx)

Function description

Get RTC_TAMP2 detection flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR TAMP2F LL_RTC_IsActiveFlag_TAMP2

LL_RTC_IsActiveFlag_TAMP1

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TAMP1 (RTC_TypeDef * RTCx)

Function description

Get RTC_TAMP1 detection flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR TAMP1F LL_RTC_IsActiveFlag_TAMP1

LL_RTC_IsActiveFlag_TSOV

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TSOV (RTC_TypeDef * RTCx)

Function description

Get Time-stamp overflow flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR TSOVF LL_RTC_IsActiveFlag_TSOV

LL_RTC_IsActiveFlag_TS

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_TS (RTC_TypeDef * RTCx)

Function description

Get Time-stamp flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR TSF LL_RTC_IsActiveFlag_TS

LL_RTC_IsActiveFlag_WUT

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_WUT (RTC_TypeDef * RTCx)

Function description

Get Wakeup timer flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR WUTF LL_RTC_IsActiveFlag_WUT

LL_RTC_IsActiveFlag_ALRB

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALRB (RTC_TypeDef * RTCx)

Function description

Get Alarm B flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR ALRBF LL_RTC_IsActiveFlag_ALRB

LL_RTC_IsActiveFlag_ALRA

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALRA (RTC_TypeDef * RTCx)

Function description

Get Alarm A flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR ALRAF LL_RTC_IsActiveFlag_ALRA

LL_RTC_ClearFlag_TAMP2

Function name

__STATIC_INLINE void LL_RTC_ClearFlag_TAMP2 (RTC_TypeDef * RTCx)

Function description

Clear RTC_TAMP2 detection flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR TAMP2F LL_RTC_ClearFlag_TAMP2

LL_RTC_ClearFlag_TAMP1

Function name

__STATIC_INLINE void LL_RTC_ClearFlag_TAMP1 (RTC_TypeDef * RTCx)

Function description

Clear RTC_TAMP1 detection flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR TAMP1F LL_RTC_ClearFlag_TAMP1

LL_RTC_ClearFlag_TSOV

Function name

__STATIC_INLINE void LL_RTC_ClearFlag_TSOV (RTC_TypeDef * RTCx)

Function description

Clear Time-stamp overflow flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR TSOVF LL_RTC_ClearFlag_TSOV

LL_RTC_ClearFlag_TS

Function name

__STATIC_INLINE void LL_RTC_ClearFlag_TS (RTC_TypeDef * RTCx)

Function description

Clear Time-stamp flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR TSF LL_RTC_ClearFlag_TS

LL_RTC_ClearFlag_WUT

Function name

__STATIC_INLINE void LL_RTC_ClearFlag_WUT (RTC_TypeDef * RTCx)

Function description

Clear Wakeup timer flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR WUTF LL_RTC_ClearFlag_WUT

LL_RTC_ClearFlag_ALRB

Function name

__STATIC_INLINE void LL_RTC_ClearFlag_ALRB (RTC_TypeDef * RTCx)

Function description

Clear Alarm B flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR ALRBF LL_RTC_ClearFlag_ALRB

LL_RTC_ClearFlag_ALRA

Function name

__STATIC_INLINE void LL_RTC_ClearFlag_ALRA (RTC_TypeDef * RTCx)

Function description

Clear Alarm A flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR ALRAF LL_RTC_ClearFlag_ALRA

LL_RTC_IsActiveFlag_INIT

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_INIT (RTC_TypeDef * RTCx)

Function description

Get Initialization flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR INITF LL_RTC_IsActiveFlag_INIT

LL_RTC_IsActiveFlag_RS

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_RS (RTC_TypeDef * RTCx)

Function description

Get Registers synchronization flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR RSF LL_RTC_IsActiveFlag_RS

LL_RTC_ClearFlag_RS

Function name

__STATIC_INLINE void LL_RTC_ClearFlag_RS (RTC_TypeDef * RTCx)

Function description

Clear Registers synchronization flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- ISR RSF LL_RTC_ClearFlag_RS

LL_RTC_IsActiveFlag_INITS

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_INITS (RTC_TypeDef * RTCx)

Function description

Get Initialization status flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR INITS LL_RTC_IsActiveFlag_INITS

LL_RTC_IsActiveFlag_SHP

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_SHP (RTC_TypeDef * RTCx)

Function description

Get Shift operation pending flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR SHPF LL_RTC_IsActiveFlag_SHP

LL_RTC_IsActiveFlag_WUTW

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_WUTW (RTC_TypeDef * RTCx)

Function description

Get Wakeup timer write flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR WUTWF LL_RTC_IsActiveFlag_WUTW

LL_RTC_IsActiveFlag_ALRBW

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALRBW (RTC_TypeDef * RTCx)

Function description

Get Alarm B write flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR ALRBWF LL_RTC_IsActiveFlag_ALRBW

LL_RTC_IsActiveFlag_ALRAW

Function name

__STATIC_INLINE uint32_t LL_RTC_IsActiveFlag_ALRAW (RTC_TypeDef * RTCx)

Function description

Get Alarm A write flag.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- ISR ALRAWF LL_RTC_IsActiveFlag_ALRAW

LL_RTC_EnableIT_TS

Function name

__STATIC_INLINE void LL_RTC_EnableIT_TS (RTC_TypeDef * RTCx)

Function description

Enable Time-stamp interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR TSIE LL_RTC_EnableIT_TS

LL_RTC_DisableIT_TS

Function name

__STATIC_INLINE void LL_RTC_DisableIT_TS (RTC_TypeDef * RTCx)

Function description

Disable Time-stamp interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR TSIE LL_RTC_DisableIT_TS

LL_RTC_EnableIT_WUT

Function name

__STATIC_INLINE void LL_RTC_EnableIT_WUT (RTC_TypeDef * RTCx)

Function description

Enable Wakeup timer interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR WUTIE LL_RTC_EnableIT_WUT

LL_RTC_DisableIT_WUT

Function name

__STATIC_INLINE void LL_RTC_DisableIT_WUT (RTC_TypeDef * RTCx)

Function description

Disable Wakeup timer interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR WUTIE LL_RTC_DisableIT_WUT

LL_RTC_EnableIT_ALRB

Function name

__STATIC_INLINE void LL_RTC_EnableIT_ALRB (RTC_TypeDef * RTCx)

Function description

Enable Alarm B interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ALRBIE LL_RTC_EnableIT_ALRB

LL_RTC_DisableIT_ALRB

Function name

__STATIC_INLINE void LL_RTC_DisableIT_ALRB (RTC_TypeDef * RTCx)

Function description

Disable Alarm B interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ALRBIE LL_RTC_DisableIT_ALRB

LL_RTC_EnableIT_ALRA

Function name

__STATIC_INLINE void LL_RTC_EnableIT_ALRA (RTC_TypeDef * RTCx)

Function description

Enable Alarm A interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ALRAIE LL_RTC_EnableIT_ALRA

LL_RTC_DisableIT_ALRA

Function name

__STATIC_INLINE void LL_RTC_DisableIT_ALRA (RTC_TypeDef * RTCx)

Function description

Disable Alarm A interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Notes

- Bit is write-protected. LL_RTC_DisableWriteProtection function should be called before.

Reference Manual to LL API cross reference:

- CR ALRAIE LL_RTC_DisableIT_ALRA

LL_RTC_EnableIT_TAMP

Function name

__STATIC_INLINE void LL_RTC_EnableIT_TAMP (RTC_TypeDef * RTCx)

Function description

Enable all Tamper Interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPIE LL_RTC_EnableIT_TAMP

LL_RTC_DisableIT_TAMP

Function name

__STATIC_INLINE void LL_RTC_DisableIT_TAMP (RTC_TypeDef * RTCx)

Function description

Disable all Tamper Interrupt.

Parameters

- **RTCx:** RTC Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- TAFCR TAMPIE LL_RTC_DisableIT_TAMP

LL_RTC_IsEnabledIT_TS

Function name

__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_TS (RTC_TypeDef * RTCx)

Function description

Check if Time-stamp interrupt is enabled or not.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR TSIE LL_RTC_IsEnabledIT_TS

LL_RTC_IsEnabledIT_WUT

Function name

__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_WUT (RTC_TypeDef * RTCx)

Function description

Check if Wakeup timer interrupt is enabled or not.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR WUTIE LL_RTC_IsEnabledIT_WUT

LL_RTC_IsEnabledIT_ALRB

Function name

__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_ALRB (RTC_TypeDef * RTCx)

Function description

Check if Alarm B interrupt is enabled or not.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR ALRBIE LL_RTC_IsEnabledIT_ALRB

LL_RTC_IsEnabledIT_ALRA

Function name

__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_ALRA (RTC_TypeDef * RTCx)

Function description

Check if Alarm A interrupt is enabled or not.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR ALRAIE LL_RTC_IsEnabledIT_ALRA

LL_RTC_IsEnabledIT_TAMP

Function name

__STATIC_INLINE uint32_t LL_RTC_IsEnabledIT_TAMP (RTC_TypeDef * RTCx)

Function description

Check if all the TAMPER interrupts are enabled or not.

Parameters

- **RTCx:** RTC Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- TAFCR TAMPIE LL_RTC_IsEnabledIT_TAMP

LL_RTC_DeInit

Function name

ErrorStatus LL_RTC_DeInit (RTC_TypeDef * RTCx)

Function description

De-Initializes the RTC registers to their default reset values.

Parameters

- **RTCx:** RTC Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RTC registers are de-initialized
  - ERROR: RTC registers are not de-initialized

Notes

- This function doesn't reset the RTC Clock source and RTC Backup Data registers.

LL_RTC_Init

Function name

ErrorStatus LL_RTC_Init (RTC_TypeDef * RTCx, LL_RTC_InitTypeDef * RTC_InitStruct)

Function description

Initializes the RTC registers according to the specified parameters in RTC_InitStruct.

Parameters

- **RTCx:** RTC Instance
- **RTC_InitStruct:** pointer to a LL_RTC_InitTypeDef structure that contains the configuration information for the RTC peripheral.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RTC registers are initialized
  - ERROR: RTC registers are not initialized

Notes

- The RTC Prescaler register is write protected and can be written in initialization mode only.

LL_RTC_StructInit

Function name

void LL_RTC_StructInit (LL_RTC_InitTypeDef * RTC_InitStruct)

Function description

Set each LL_RTC_InitTypeDef field to default value.

Parameters

- **RTC_InitStruct:** pointer to a LL_RTC_InitTypeDef structure which will be initialized.

Return values

- **None:**

LL_RTC_TIME_Init

Function name

ErrorStatus LL_RTC_TIME_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_TimeTypeDef * RTC_TimeStruct)

Function description

Set the RTC current time.

Parameters

- **RTCx:** RTC Instance
- **RTC_Format:** This parameter can be one of the following values:
  - LL_RTC_FORMAT_BIN
  - LL_RTC_FORMAT_BCD
- **RTC_TimeStruct:** pointer to a RTC_TimeTypeDef structure that contains the time configuration information for the RTC.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RTC Time register is configured
  - ERROR: RTC Time register is not configured

LL_RTC_TIME_StructInit

Function name

void LL_RTC_TIME_StructInit (LL_RTC_TimeTypeDef * RTC_TimeStruct)

Function description

Set each LL_RTC_TimeTypeDef field to default value (Time = 00h:00min:00sec).

Parameters

- **RTC_TimeStruct:** pointer to a LL_RTC_TimeTypeDef structure which will be initialized.

Return values

- **None:**

LL_RTC_DATE_Init

Function name

ErrorStatus LL_RTC_DATE_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_DateTypeDef * RTC_DateStruct)

Function description

Set the RTC current date.

Parameters

- **RTCx:** RTC Instance
- **RTC_Format:** This parameter can be one of the following values:
  - LL_RTC_FORMAT_BIN
  - LL_RTC_FORMAT_BCD
- **RTC_DateStruct:** pointer to a RTC_DateTypeDef structure that contains the date configuration information for the RTC.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RTC Day register is configured
  - ERROR: RTC Day register is not configured

LL_RTC_DATE_StructInit

Function name

void LL_RTC_DATE_StructInit (LL_RTC_DateTypeDef * RTC_DateStruct)

Function description

Set each LL_RTC_DateTypeDef field to default value (date = Monday, January 01 xx00)

Parameters

- **RTC_DateStruct:** pointer to a LL_RTC_DateTypeDef structure which will be initialized.

Return values

- **None:**

LL_RTC_ALMA_Init

Function name

ErrorStatus LL_RTC_ALMA_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_AlarmTypeDef * RTC_AlarmStruct)

Function description

Set the RTC Alarm A.

Parameters

- **RTCx:** RTC Instance
- **RTC_Format:** This parameter can be one of the following values:
  - LL_RTC_FORMAT_BIN
  - LL_RTC_FORMAT_BCD
- **RTC_AlarmStruct:** pointer to a LL_RTC_AlarmTypeDef structure that contains the alarm configuration parameters.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: ALARMA registers are configured
  - ERROR: ALARMA registers are not configured

Notes

- The Alarm register can only be written when the corresponding Alarm is disabled (Use LL_RTC_ALMA_Disable function).

LL_RTC_ALMB_Init

Function name

ErrorStatus LL_RTC_ALMB_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_AlarmTypeDef * RTC_AlarmStruct)

Function description

Set the RTC Alarm B.

Parameters

- **RTCx:** RTC Instance
- **RTC_Format:** This parameter can be one of the following values:
  - LL_RTC_FORMAT_BIN
  - LL_RTC_FORMAT_BCD
- **RTC_AlarmStruct:** pointer to a LL_RTC_AlarmTypeDef structure that contains the alarm configuration parameters.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: ALARMB registers are configured
  - ERROR: ALARMB registers are not configured

Notes

- The Alarm register can only be written when the corresponding Alarm is disabled (LL_RTC_ALMB_Disable function).

LL_RTC_ALMA_StructInit

Function name

void LL_RTC_ALMA_StructInit (LL_RTC_AlarmTypeDef * RTC_AlarmStruct)

Function description

Set each LL_RTC_AlarmTypeDef of ALARMA field to default value (Time = 00h:00mn:00sec / Day = 1st day of the month/Mask = all fields are masked).

Parameters

- **RTC_AlarmStruct:** pointer to a LL_RTC_AlarmTypeDef structure which will be initialized.

Return values

- **None:**

LL_RTC_ALMB_StructInit

Function name

void LL_RTC_ALMB_StructInit (LL_RTC_AlarmTypeDef * RTC_AlarmStruct)

Function description

Set each LL_RTC_AlarmTypeDef of ALARMA field to default value (Time = 00h:00mn:00sec / Day = 1st day of the month/Mask = all fields are masked).

Parameters

- **RTC_AlarmStruct:** pointer to a LL_RTC_AlarmTypeDef structure which will be initialized.

Return values

- **None:**

LL_RTC_EnterInitMode

Function name

ErrorStatus LL_RTC_EnterInitMode (RTC_TypeDef * RTCx)

Function description

Enters the RTC Initialization mode.

Parameters

- **RTCx:** RTC Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RTC is in Init mode
  - ERROR: RTC is not in Init mode

Notes

- The RTC Initialization mode is write protected, use the LL_RTC_DisableWriteProtection before calling this function.

LL_RTC_ExitInitMode

Function name

ErrorStatus LL_RTC_ExitInitMode (RTC_TypeDef * RTCx)

Function description

Exit the RTC Initialization mode.

Parameters

- **RTCx:** RTC Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RTC exited from in Init mode
  - ERROR: Not applicable

Notes

- When the initialization sequence is complete, the calendar restarts counting after 4 RTCCLK cycles.
- The RTC Initialization mode is write protected, use the LL_RTC_DisableWriteProtection before calling this function.

LL_RTC_WaitForSynchro

Function name

ErrorStatus LL_RTC_WaitForSynchro (RTC_TypeDef * RTCx)

Function description

Waits until the RTC Time and Day registers (RTC_TR and RTC_DR) are synchronized with RTC APB clock.

Parameters

- **RTCx:** RTC Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: RTC registers are synchronised
  - ERROR: RTC registers are not synchronised

Notes

- The RTC Resynchronization mode is write protected, use the LL_RTC_DisableWriteProtection before calling this function.
- To read the calendar through the shadow registers after Calendar initialization, calendar update or after wakeup from low power modes the software must first clear the RSF flag. The software must then wait until it is set again before reading the calendar, which means that the calendar registers have been correctly copied into the RTC_TR and RTC_DR shadow registers.

89.3 RTC Firmware driver defines

The following section lists the various define and macros of the module.

89.3.1 RTC

RTC

ALARM OUTPUT

LL_RTC_ALARMOUT_DISABLE

Output disabled

LL_RTC_ALARMOUT_ALMA

Alarm A output enabled

LL_RTC_ALARMOUT_ALMB

Alarm B output enabled

LL_RTC_ALARMOUT_WAKEUP

Wakeup output enabled

ALARM OUTPUT TYPE

LL_RTC_ALARM_OUTPUTTYPE_OPENDRAIN

RTC_ALARM, when mapped on PC13, is open-drain output

LL_RTC_ALARM_OUTPUTTYPE_PUSHPULL

RTC_ALARM, when mapped on PC13, is push-pull output

ALARMA MASK

LL_RTC_ALMA_MASK_NONE

No masks applied on Alarm A

LL_RTC_ALMA_MASK_DATEWEEKDAY

Date/day do not care in Alarm A comparison

LL_RTC_ALMA_MASK_HOURS

Hours do not care in Alarm A comparison

LL_RTC_ALMA_MASK_MINUTES

Minutes do not care in Alarm A comparison

LL_RTC_ALMA_MASK_SECONDS

Seconds do not care in Alarm A comparison

LL_RTC_ALMA_MASK_ALL

Masks all

ALARMA TIME FORMAT

LL_RTC_ALMA_TIME_FORMAT_AM

AM or 24-hour format

LL_RTC_ALMA_TIME_FORMAT_PM

PM

RTC Alarm A Date WeekDay

LL_RTC_ALMA_DATEWEEKDAYSEL_DATE

Alarm A Date is selected

LL_RTC_ALMA_DATEWEEKDAYSEL_WEEKDAY

Alarm A WeekDay is selected

ALARMB MASK

LL_RTC_ALMB_MASK_NONE

No masks applied on Alarm B

LL_RTC_ALMB_MASK_DATEWEEKDAY

Date/day do not care in Alarm B comparison

LL_RTC_ALMB_MASK_HOURS

Hours do not care in Alarm B comparison

LL_RTC_ALMB_MASK_MINUTES

Minutes do not care in Alarm B comparison

LL_RTC_ALMB_MASK_SECONDS

Seconds do not care in Alarm B comparison

LL_RTC_ALMB_MASK_ALL

Masks all

ALARMB TIME FORMAT

LL_RTC_ALMB_TIME_FORMAT_AM

AM or 24-hour format

LL_RTC_ALMB_TIME_FORMAT_PM

PM

RTC Alarm B Date WeekDay

LL_RTC_ALMB_DATEWEEKDAYSEL_DATE

Alarm B Date is selected

LL_RTC_ALMB_DATEWEEKDAYSEL_WEEKDAY

Alarm B WeekDay is selected

BACKUP

LL_RTC_BKP_DR0

LL_RTC_BKP_DR1

LL_RTC_BKP_DR2

LL_RTC_BKP_DR3

LL_RTC_BKP_DR4

LL_RTC_BKP_DR5

LL_RTC_BKP_DR6

LL_RTC_BKP_DR7

LL_RTC_BKP_DR8

LL_RTC_BKP_DR9

LL_RTC_BKP_DR10

LL_RTC_BKP_DR11

LL_RTC_BKP_DR12

LL_RTC_BKP_DR13

LL_RTC_BKP_DR14

LL_RTC_BKP_DR15

LL_RTC_BKP_DR16

LL_RTC_BKP_DR17

LL_RTC_BKP_DR18

LL_RTC_BKP_DR19

Calibration pulse insertion

LL_RTC_CALIB_INSERTPULSE_NONE

No RTCCLK pulses are added

LL_RTC_CALIB_INSERTPULSE_SET

One RTCCLK pulse is effectively inserted every 2exp11 pulses (frequency increased by 488.5 ppm)

Calibration output

LL_RTC_CALIB_OUTPUT_NONE

Calibration output disabled

LL_RTC_CALIB_OUTPUT_1HZ

Calibration output is 1 Hz

LL_RTC_CALIB_OUTPUT_512HZ

Calibration output is 512 Hz

Calibration period

LL_RTC_CALIB_PERIOD_32SEC

Use a 32-second calibration cycle period

LL_RTC_CALIB_PERIOD_16SEC

Use a 16-second calibration cycle period

LL_RTC_CALIB_PERIOD_8SEC

Use a 8-second calibration cycle period

Coarse digital calibration sign

LL_RTC_CALIB_SIGN_POSITIVE

Positive calibration: calendar update frequency is increased

LL_RTC_CALIB_SIGN_NEGATIVE

Negative calibration: calendar update frequency is decreased

FORMAT

LL_RTC_FORMAT_BIN

Binary data format

LL_RTC_FORMAT_BCD

BCD data format

Get Flags Defines

LL_RTC_ISR_RECALPF

LL_RTC_ISR_TAMP3F

LL_RTC_ISR_TAMP2F

LL_RTC_ISR_TAMP1F

LL_RTC_ISR_TSOVF

LL_RTC_ISR_TSF

LL_RTC_ISR_WUTF

LL_RTC_ISR_ALRBF

LL_RTC_ISR_ALRAF

LL_RTC_ISR_INITF

LL_RTC_ISR_RSF

LL_RTC_ISR_INITS

LL_RTC_ISR_SHPF

LL_RTC_ISR_WUTWF

LL_RTC_ISR_ALRBWF

LL_RTC_ISR_ALRAWF

HOUR FORMAT

LL_RTC_HOURFORMAT_24HOUR

24 hour/day format

LL_RTC_HOURFORMAT_AMPM

AM/PM hour format

IT Defines

LL_RTC_CR_TSIE

LL_RTC_CR_WUTIE

LL_RTC_CR_ALRBIE

LL_RTC_CR_ALRAIE

LL_RTC_TAFCR_TAMPIE

MONTH

LL_RTC_MONTH_JANUARY

January

LL_RTC_MONTH_FEBRUARY

February

LL_RTC_MONTH_MARCH

March

LL_RTC_MONTH_APRIL

April

LL_RTC_MONTH_MAY

May

LL_RTC_MONTH_JUNE

June

LL_RTC_MONTH_JULY

July

LL_RTC_MONTH_AUGUST

August

LL_RTC_MONTH_SEPTEMBER

September

LL_RTC_MONTH_OCTOBER

October

LL_RTC_MONTH_NOVEMBER

November

LL_RTC_MONTH_DECEMBER

December

OUTPUT POLARITY PIN

LL_RTC_OUTPUTPOLARITY_PIN_HIGH

Pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL)

LL_RTC_OUTPUTPOLARITY_PIN_LOW

Pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL)

PIN

LL_RTC_PIN_PC13

PC13 is forced to push-pull output if all RTC alternate functions are disabled

LL_RTC_PIN_PC14

PC14 is forced to push-pull output if LSE is disabled

LL_RTC_PIN_PC15

PC15 is forced to push-pull output if LSE is disabled

SHIFT SECOND

LL_RTC_SHIFT_SECOND_DELAY

LL_RTC_SHIFT_SECOND_ADVANCE

TAMPER1 mapping

LL_RTC_TamperPin_Default

Use RTC_AF1 as TAMPER1

LL_RTC_TamperPin_Pos1

Use RTC_AF2 as TAMPER1

TAMPER

LL_RTC_TAMPER_1

RTC_TAMP1 input detection

LL_RTC_TAMPER_2

RTC_TAMP2 input detection

TAMPER ACTIVE LEVEL

LL_RTC_TAMPER_ACTIVELEVEL_TAMP1

RTC_TAMP1 input falling edge (if TAMPFLT = 00) or staying high (if TAMPFLT != 00) triggers a tamper detection event

LL_RTC_TAMPER_ACTIVELEVEL_TAMP2

RTC_TAMP2 input falling edge (if TAMPFLT = 00) or staying high (if TAMPFLT != 00) triggers a tamper detection event

TAMPER DURATION

LL_RTC_TAMPER_DURATION_1RTCCLK

Tamper pins are pre-charged before sampling during 1 RTCCLK cycle

LL_RTC_TAMPER_DURATION_2RTCCLK

Tamper pins are pre-charged before sampling during 2 RTCCLK cycles

LL_RTC_TAMPER_DURATION_4RTCCLK

Tamper pins are pre-charged before sampling during 4 RTCCLK cycles

LL_RTC_TAMPER_DURATION_8RTCCLK

Tamper pins are pre-charged before sampling during 8 RTCCLK cycles

TAMPER FILTER

LL_RTC_TAMPER_FILTER_DISABLE

Tamper filter is disabled

LL_RTC_TAMPER_FILTER_2SAMPLE

Tamper is activated after 2 consecutive samples at the active level

LL_RTC_TAMPER_FILTER_4SAMPLE

Tamper is activated after 4 consecutive samples at the active level

LL_RTC_TAMPER_FILTER_8SAMPLE

Tamper is activated after 8 consecutive samples at the active level.

TAMPER MASK

LL_RTC_TAMPER_MASK_TAMPER1

Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware.The backup registers are not erased

LL_RTC_TAMPER_MASK_TAMPER2

Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers are not erased.

TAMPER NO ERASE

LL_RTC_TAMPER_NOERASE_TAMPER1

Tamper 1 event does not erase the backup registers.

LL_RTC_TAMPER_NOERASE_TAMPER2

Tamper 2 event does not erase the backup registers.

TAMPER SAMPLING FREQUENCY DIVIDER

LL_RTC_TAMPER_SAMPLFREQDIV_32768

Each of the tamper inputs are sampled with a frequency = RTCCLK / 32768

LL_RTC_TAMPER_SAMPLFREQDIV_16384

Each of the tamper inputs are sampled with a frequency = RTCCLK / 16384

LL_RTC_TAMPER_SAMPLFREQDIV_8192

Each of the tamper inputs are sampled with a frequency = RTCCLK / 8192

LL_RTC_TAMPER_SAMPLFREQDIV_4096

Each of the tamper inputs are sampled with a frequency = RTCCLK / 4096

LL_RTC_TAMPER_SAMPLFREQDIV_2048

Each of the tamper inputs are sampled with a frequency = RTCCLK / 2048

LL_RTC_TAMPER_SAMPLFREQDIV_1024

Each of the tamper inputs are sampled with a frequency = RTCCLK / 1024

LL_RTC_TAMPER_SAMPLFREQDIV_512

Each of the tamper inputs are sampled with a frequency = RTCCLK / 512

LL_RTC_TAMPER_SAMPLFREQDIV_256

Each of the tamper inputs are sampled with a frequency = RTCCLK / 256

TIMESTAMP EDGE

LL_RTC_TIMESTAMP_EDGE_RISING

RTC_TS input rising edge generates a time-stamp event

LL_RTC_TIMESTAMP_EDGE_FALLING

RTC_TS input falling edge generates a time-stamp even

TIME FORMAT

LL_RTC_TIME_FORMAT_AM_OR_24

AM or 24-hour format

LL_RTC_TIME_FORMAT_PM

PM

TIMESTAMP mapping

LL_RTC_TimeStampPin_Default

Use RTC_AF1 as TIMESTAMP

LL_RTC_TimeStampPin_Pos1

Use RTC_AF2 as TIMESTAMP

TIMESTAMP TIME FORMAT

LL_RTC_TS_TIME_FORMAT_AM

AM or 24-hour format

LL_RTC_TS_TIME_FORMAT_PM

PM

WAKEUP CLOCK DIV

LL_RTC_WAKEUPCLOCK_DIV_16

RTC/16 clock is selected

LL_RTC_WAKEUPCLOCK_DIV_8

RTC/8 clock is selected

LL_RTC_WAKEUPCLOCK_DIV_4

RTC/4 clock is selected

LL_RTC_WAKEUPCLOCK_DIV_2

RTC/2 clock is selected

LL_RTC_WAKEUPCLOCK_CKSPRE

ck_spre (usually 1 Hz) clock is selected

LL_RTC_WAKEUPCLOCK_CKSPRE_WUT

ck_spre (usually 1 Hz) clock is selected and 2exp16 is added to the WUT counter value

WEEK DAY

LL_RTC_WEEKDAY_MONDAY

Monday

LL_RTC_WEEKDAY_TUESDAY

Tuesday

LL_RTC_WEEKDAY_WEDNESDAY

Wednesday

LL_RTC_WEEKDAY_THURSDAY

Thrusday

LL_RTC_WEEKDAY_FRIDAY

Friday

LL_RTC_WEEKDAY_SATURDAY

Saturday

LL_RTC_WEEKDAY_SUNDAY

Sunday

Convert helper Macros

__LL_RTC_CONVERT_BIN2BCD

Description:

- Helper macro to convert a value from 2 digit decimal format to BCD format.

Parameters:

- __VALUE__: Byte to be converted

Return value:

- Converted: byte

__LL_RTC_CONVERT_BCD2BIN

Description:

- Helper macro to convert a value from BCD format to 2 digit decimal format.

Parameters:

- __VALUE__: BCD value to be converted

Return value:

- Converted: byte

Date helper Macros

__LL_RTC_GET_WEEKDAY

Description:

- Helper macro to retrieve weekday.

Parameters:

- __RTC_DATE__: Date returned by

Return value:

- Returned: value can be one of the following values:
  - LL_RTC_WEEKDAY_MONDAY
  - LL_RTC_WEEKDAY_TUESDAY
  - LL_RTC_WEEKDAY_WEDNESDAY
  - LL_RTC_WEEKDAY_THURSDAY
  - LL_RTC_WEEKDAY_FRIDAY
  - LL_RTC_WEEKDAY_SATURDAY
  - LL_RTC_WEEKDAY_SUNDAY

__LL_RTC_GET_YEAR

Description:

- Helper macro to retrieve Year in BCD format.

Parameters:

- __RTC_DATE__: Value returned by

Return value:

- Year: in BCD format (0x00 . . . 0x99)

__LL_RTC_GET_MONTH

Description:

- Helper macro to retrieve Month in BCD format.

Parameters:

- __RTC_DATE__: Value returned by

Return value:

- Returned: value can be one of the following values:
  - LL_RTC_MONTH_JANUARY
  - LL_RTC_MONTH_FEBRUARY
  - LL_RTC_MONTH_MARCH
  - LL_RTC_MONTH_APRIL
  - LL_RTC_MONTH_MAY
  - LL_RTC_MONTH_JUNE
  - LL_RTC_MONTH_JULY
  - LL_RTC_MONTH_AUGUST
  - LL_RTC_MONTH_SEPTEMBER
  - LL_RTC_MONTH_OCTOBER
  - LL_RTC_MONTH_NOVEMBER
  - LL_RTC_MONTH_DECEMBER

__LL_RTC_GET_DAY

Description:

- Helper macro to retrieve Day in BCD format.

Parameters:

- __RTC_DATE__: Value returned by

Return value:

- Day: in BCD format (0x01 . . . 0x31)

Time helper Macros

__LL_RTC_GET_HOUR

Description:

- Helper macro to retrieve hour in BCD format.

Parameters:

- __RTC_TIME__: RTC time returned by

Return value:

- Hours: in BCD format (0x01. . .0x12 or between Min_Data=0x00 and Max_Data=0x23)

__LL_RTC_GET_MINUTE

Description:

- Helper macro to retrieve minute in BCD format.

Parameters:

- __RTC_TIME__: RTC time returned by

Return value:

- Minutes: in BCD format (0x00. . .0x59)

__LL_RTC_GET_SECOND

Description:

- Helper macro to retrieve second in BCD format.

Parameters:

- __RTC_TIME__: RTC time returned by

Return value:

- Seconds: in format (0x00. . .0x59)

Common Write and read registers Macros

LL_RTC_WriteReg

Description:

- Write a value in RTC register.

Parameters:

- __INSTANCE__: RTC Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_RTC_ReadReg

Description:

- Read a value in RTC register.

Parameters:

- __INSTANCE__: RTC Instance
- __REG__: Register to be read

Return value:

- Register: value

90 LL SPI Generic Driver

90.1 SPI Firmware driver registers structures

90.1.1 LL_SPI_InitTypeDef

LL_SPI_InitTypeDef is defined in the stm32f4xx_ll_spi.h

Data Fields

- ***uint32_t TransferDirection***
- ***uint32_t Mode***
- ***uint32_t DataWidth***
- ***uint32_t ClockPolarity***
- ***uint32_t ClockPhase***
- ***uint32_t NSS***
- ***uint32_t BaudRate***
- ***uint32_t BitOrder***
- ***uint32_t CRCCalculation***
- ***uint32_t CRCPoly***

Field Documentation

- ***uint32_t LL_SPI_InitTypeDef::TransferDirection***

  Specifies the SPI unidirectional or bidirectional data mode. This parameter can be a value of ***SPI_LL_EC_TRANSFER_MODE***.This feature can be modified afterwards using unitary function **LL_SPI_SetTransferDirection()**.
- ***uint32_t LL_SPI_InitTypeDef::Mode***

  Specifies the SPI mode (Master/Slave). This parameter can be a value of ***SPI_LL_EC_MODE***.This feature can be modified afterwards using unitary function **LL_SPI_SetMode()**.
- ***uint32_t LL_SPI_InitTypeDef::DataWidth***

  Specifies the SPI data width. This parameter can be a value of ***SPI_LL_EC_DATAWIDTH***.This feature can be modified afterwards using unitary function **LL_SPI_SetDataWidth()**.
- ***uint32_t LL_SPI_InitTypeDef::ClockPolarity***

  Specifies the serial clock steady state. This parameter can be a value of ***SPI_LL_EC_POLARITY***.This feature can be modified afterwards using unitary function **LL_SPI_SetClockPolarity()**.
- ***uint32_t LL_SPI_InitTypeDef::ClockPhase***

  Specifies the clock active edge for the bit capture. This parameter can be a value of ***SPI_LL_EC_PHASE***.This feature can be modified afterwards using unitary function **LL_SPI_SetClockPhase()**.
- ***uint32_t LL_SPI_InitTypeDef::NSS***

  Specifies whether the NSS signal is managed by hardware (NSS pin) or by software using the SSI bit. This parameter can be a value of ***SPI_LL_EC_NSS_MODE***.This feature can be modified afterwards using unitary function **LL_SPI_SetNSSMode()**.
- ***uint32_t LL_SPI_InitTypeDef::BaudRate***

  Specifies the BaudRate prescaler value which will be used to configure the transmit and receive SCK clock. This parameter can be a value of ***SPI_LL_EC_BAUDRATEPRESCALER***.

  Note:

  – The communication clock is derived from the master clock. The slave clock does not need to be set.

  This feature can be modified afterwards using unitary function **LL_SPI_SetBaudRatePrescaler()**.
- ***uint32_t LL_SPI_InitTypeDef::BitOrder***

  Specifies whether data transfers start from MSB or LSB bit. This parameter can be a value of ***SPI_LL_EC_BIT_ORDER***.This feature can be modified afterwards using unitary function **LL_SPI_SetTransferBitOrder()**.

- ***uint32_t LL_SPI_InitTypeDef::CRCCalculation***

  Specifies if the CRC calculation is enabled or not. This parameter can be a value of ***SPI_LL_EC_CRC_CALCULATION***.This feature can be modified afterwards using unitary functions **LL_SPI_EnableCRC()** and **LL_SPI_DisableCRC()**.
- ***uint32_t LL_SPI_InitTypeDef::CRCPoly***

  Specifies the polynomial used for the CRC calculation. This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFFFF.This feature can be modified afterwards using unitary function **LL_SPI_SetCRCPolynomial()**.

90.1.2 LL_I2S_InitTypeDef

LL_I2S_InitTypeDef is defined in the stm32f4xx_ll_spi.h

Data Fields

- ***uint32_t Mode***
- ***uint32_t Standard***
- ***uint32_t DataFormat***
- ***uint32_t MCLKOutput***
- ***uint32_t AudioFreq***
- ***uint32_t ClockPolarity***

Field Documentation

- ***uint32_t LL_I2S_InitTypeDef::Mode***

  Specifies the I2S operating mode. This parameter can be a value of ***I2S_LL_EC_MODE***This feature can be modified afterwards using unitary function **LL_I2S_SetTransferMode()**.
- ***uint32_t LL_I2S_InitTypeDef::Standard***

  Specifies the standard used for the I2S communication. This parameter can be a value of ***I2S_LL_EC_STANDARD***This feature can be modified afterwards using unitary function **LL_I2S_SetStandard()**.
- ***uint32_t LL_I2S_InitTypeDef::DataFormat***

  Specifies the data format for the I2S communication. This parameter can be a value of ***I2S_LL_EC_DATA_FORMAT***This feature can be modified afterwards using unitary function **LL_I2S_SetDataFormat()**.
- ***uint32_t LL_I2S_InitTypeDef::MCLKOutput***

  Specifies whether the I2S MCLK output is enabled or not. This parameter can be a value of ***I2S_LL_EC_MCLK_OUTPUT***This feature can be modified afterwards using unitary functions **LL_I2S_EnableMasterClock()** or **LL_I2S_DisableMasterClock**.
- ***uint32_t LL_I2S_InitTypeDef::AudioFreq***

  Specifies the frequency selected for the I2S communication. This parameter can be a value of ***I2S_LL_EC_AUDIO_FREQ***Audio Frequency can be modified afterwards using Reference manual formulas to calculate Prescaler Linear, Parity and unitary functions **LL_I2S_SetPrescalerLinear()** and **LL_I2S_SetPrescalerParity()** to set it.
- ***uint32_t LL_I2S_InitTypeDef::ClockPolarity***

  Specifies the idle state of the I2S clock. This parameter can be a value of ***I2S_LL_EC_POLARITY***This feature can be modified afterwards using unitary function **LL_I2S_SetClockPolarity()**.

90.2 SPI Firmware driver API description

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

90.2.1 Detailed description of functions

LL_SPI_Enable

Function name

__STATIC_INLINE void LL_SPI_Enable (SPI_TypeDef * SPIx)

Function description

Enable SPI peripheral.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 SPE LL_SPI_Enable

LL_SPI_Disable

Function name

__STATIC_INLINE void LL_SPI_Disable (SPI_TypeDef * SPIx)

Function description

Disable SPI peripheral.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- When disabling the SPI, follow the procedure described in the Reference Manual.

Reference Manual to LL API cross reference:

- CR1 SPE LL_SPI_Disable

LL_SPI_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_SPI_IsEnabled (SPI_TypeDef * SPIx)

Function description

Check if SPI peripheral is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 SPE LL_SPI_IsEnabled

LL_SPI_SetMode

Function name

__STATIC_INLINE void LL_SPI_SetMode (SPI_TypeDef * SPIx, uint32_t Mode)

Function description

Set SPI operation mode to Master or Slave.

Parameters

- **SPIx:** SPI Instance
- **Mode:** This parameter can be one of the following values:
  - LL_SPI_MODE_MASTER
  - LL_SPI_MODE_SLAVE

Return values

- **None:**

Notes

- This bit should not be changed when communication is ongoing.

Reference Manual to LL API cross reference:

- CR1 MSTR LL_SPI_SetMode
- CR1 SSI LL_SPI_SetMode

LL_SPI_GetMode

Function name

__STATIC_INLINE uint32_t LL_SPI_GetMode (SPI_TypeDef * SPIx)

Function description

Get SPI operation mode (Master or Slave)

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_MODE_MASTER
  - LL_SPI_MODE_SLAVE

Reference Manual to LL API cross reference:

- CR1 MSTR LL_SPI_GetMode
- CR1 SSI LL_SPI_GetMode

LL_SPI_SetStandard

Function name

__STATIC_INLINE void LL_SPI_SetStandard (SPI_TypeDef * SPIx, uint32_t Standard)

Function description

Set serial protocol used.

Parameters

- **SPIx:** SPI Instance
- **Standard:** This parameter can be one of the following values:
  - LL_SPI_PROTOCOL_MOTOROLA
  - LL_SPI_PROTOCOL_TI

Return values

- **None:**

Notes

- This bit should be written only when SPI is disabled (SPE = 0) for correct operation.

Reference Manual to LL API cross reference:

- CR2 FRF LL_SPI_SetStandard

LL_SPI_GetStandard

Function name

__STATIC_INLINE uint32_t LL_SPI_GetStandard (SPI_TypeDef * SPIx)

Function description

Get serial protocol used.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_PROTOCOL_MOTOROLA
  - LL_SPI_PROTOCOL_TI

Reference Manual to LL API cross reference:

- CR2 FRF LL_SPI_GetStandard

LL_SPI_SetClockPhase

Function name

__STATIC_INLINE void LL_SPI_SetClockPhase (SPI_TypeDef * SPIx, uint32_t ClockPhase)

Function description

Set clock phase.

Parameters

- **SPIx:** SPI Instance
- **ClockPhase:** This parameter can be one of the following values:
  - LL_SPI_PHASE_1EDGE
  - LL_SPI_PHASE_2EDGE

Return values

- **None:**

Notes

- This bit should not be changed when communication is ongoing. This bit is not used in SPI TI mode.

Reference Manual to LL API cross reference:

- CR1 CPHA LL_SPI_SetClockPhase

LL_SPI_GetClockPhase

Function name

__STATIC_INLINE uint32_t LL_SPI_GetClockPhase (SPI_TypeDef * SPIx)

Function description

Get clock phase.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_PHASE_1EDGE
  - LL_SPI_PHASE_2EDGE

Reference Manual to LL API cross reference:

- CR1 CPHA LL_SPI_GetClockPhase

LL_SPI_SetClockPolarity

Function name

__STATIC_INLINE void LL_SPI_SetClockPolarity (SPI_TypeDef * SPIx, uint32_t ClockPolarity)

Function description

Set clock polarity.

Parameters

- **SPIx:** SPI Instance
- **ClockPolarity:** This parameter can be one of the following values:
  - LL_SPI_POLARITY_LOW
  - LL_SPI_POLARITY_HIGH

Return values

- **None:**

Notes

- This bit should not be changed when communication is ongoing. This bit is not used in SPI TI mode.

Reference Manual to LL API cross reference:

- CR1 CPOL LL_SPI_SetClockPolarity

LL_SPI_GetClockPolarity

Function name

__STATIC_INLINE uint32_t LL_SPI_GetClockPolarity (SPI_TypeDef * SPIx)

Function description

Get clock polarity.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_POLARITY_LOW
  - LL_SPI_POLARITY_HIGH

Reference Manual to LL API cross reference:

- CR1 CPOL LL_SPI_GetClockPolarity

LL_SPI_SetBaudRatePrescaler

Function name

__STATIC_INLINE void LL_SPI_SetBaudRatePrescaler (SPI_TypeDef * SPIx, uint32_t BaudRate)

Function description

Set baud rate prescaler.

Parameters

- **SPIx:** SPI Instance
- **BaudRate:** This parameter can be one of the following values:
  - LL_SPI_BAUDRATEPRESCALER_DIV2
  - LL_SPI_BAUDRATEPRESCALER_DIV4
  - LL_SPI_BAUDRATEPRESCALER_DIV8
  - LL_SPI_BAUDRATEPRESCALER_DIV16
  - LL_SPI_BAUDRATEPRESCALER_DIV32
  - LL_SPI_BAUDRATEPRESCALER_DIV64
  - LL_SPI_BAUDRATEPRESCALER_DIV128
  - LL_SPI_BAUDRATEPRESCALER_DIV256

Return values

- **None:**

Notes

- These bits should not be changed when communication is ongoing. SPI BaudRate = fPCLK/Prescaler.

Reference Manual to LL API cross reference:

- CR1 BR LL_SPI_SetBaudRatePrescaler

LL_SPI_GetBaudRatePrescaler

Function name

__STATIC_INLINE uint32_t LL_SPI_GetBaudRatePrescaler (SPI_TypeDef * SPIx)

Function description

Get baud rate prescaler.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_BAUDRATEPRESCALER_DIV2
  - LL_SPI_BAUDRATEPRESCALER_DIV4
  - LL_SPI_BAUDRATEPRESCALER_DIV8
  - LL_SPI_BAUDRATEPRESCALER_DIV16
  - LL_SPI_BAUDRATEPRESCALER_DIV32
  - LL_SPI_BAUDRATEPRESCALER_DIV64
  - LL_SPI_BAUDRATEPRESCALER_DIV128
  - LL_SPI_BAUDRATEPRESCALER_DIV256

Reference Manual to LL API cross reference:

- CR1 BR LL_SPI_GetBaudRatePrescaler

LL_SPI_SetTransferBitOrder

Function name

__STATIC_INLINE void LL_SPI_SetTransferBitOrder (SPI_TypeDef * SPIx, uint32_t BitOrder)

Function description

Set transfer bit order.

Parameters

- **SPIx:** SPI Instance
- **BitOrder:** This parameter can be one of the following values:
  - LL_SPI_LSB_FIRST
  - LL_SPI_MSB_FIRST

Return values

- **None:**

Notes

- This bit should not be changed when communication is ongoing. This bit is not used in SPI TI mode.

Reference Manual to LL API cross reference:

- CR1 LSBFIRST LL_SPI_SetTransferBitOrder

LL_SPI_GetTransferBitOrder

Function name

__STATIC_INLINE uint32_t LL_SPI_GetTransferBitOrder (SPI_TypeDef * SPIx)

Function description

Get transfer bit order.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_LSB_FIRST
  - LL_SPI_MSB_FIRST

Reference Manual to LL API cross reference:

- CR1 LSBFIRST LL_SPI_GetTransferBitOrder

LL_SPI_SetTransferDirection

Function name

__STATIC_INLINE void LL_SPI_SetTransferDirection (SPI_TypeDef * SPIx, uint32_t TransferDirection)

Function description

Set transfer direction mode.

Parameters

- **SPIx:** SPI Instance
- **TransferDirection:** This parameter can be one of the following values:
  - LL_SPI_FULL_DUPLEX
  - LL_SPI_SIMPLEX_RX
  - LL_SPI_HALF_DUPLEX_RX
  - LL_SPI_HALF_DUPLEX_TX

Return values

- **None:**

Notes

- For Half-Duplex mode, Rx Direction is set by default. In master mode, the MOSI pin is used and in slave mode, the MISO pin is used for Half-Duplex.

Reference Manual to LL API cross reference:

- CR1 RXONLY LL_SPI_SetTransferDirection
- CR1 BIDIMODE LL_SPI_SetTransferDirection
- CR1 BIDIOE LL_SPI_SetTransferDirection

LL_SPI_GetTransferDirection

Function name

__STATIC_INLINE uint32_t LL_SPI_GetTransferDirection (SPI_TypeDef * SPIx)

Function description

Get transfer direction mode.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_FULL_DUPLEX
  - LL_SPI_SIMPLEX_RX
  - LL_SPI_HALF_DUPLEX_RX
  - LL_SPI_HALF_DUPLEX_TX

Reference Manual to LL API cross reference:

- CR1 RXONLY LL_SPI_GetTransferDirection
- CR1 BIDIMODE LL_SPI_GetTransferDirection
- CR1 BIDIOE LL_SPI_GetTransferDirection

LL_SPI_SetDataWidth

Function name

__STATIC_INLINE void LL_SPI_SetDataWidth (SPI_TypeDef * SPIx, uint32_t DataWidth)

Function description

Set frame data width.

Parameters

- **SPIx:** SPI Instance
- **DataWidth:** This parameter can be one of the following values:
  - LL_SPI_DATAWIDTH_8BIT
  - LL_SPI_DATAWIDTH_16BIT

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 DFF LL_SPI_SetDataWidth

LL_SPI_GetDataWidth

Function name

__STATIC_INLINE uint32_t LL_SPI_GetDataWidth (SPI_TypeDef * SPIx)

Function description

Get frame data width.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_DATAWIDTH_8BIT
  - LL_SPI_DATAWIDTH_16BIT

Reference Manual to LL API cross reference:

- CR1 DFF LL_SPI_GetDataWidth

LL_SPI_EnableCRC

Function name

__STATIC_INLINE void LL_SPI_EnableCRC (SPI_TypeDef * SPIx)

Function description

Enable CRC.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- This bit should be written only when SPI is disabled (SPE = 0) for correct operation.

Reference Manual to LL API cross reference:

- CR1 CRCEN LL_SPI_EnableCRC

LL_SPI_DisableCRC

Function name

__STATIC_INLINE void LL_SPI_DisableCRC (SPI_TypeDef * SPIx)

Function description

Disable CRC.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- This bit should be written only when SPI is disabled (SPE = 0) for correct operation.

Reference Manual to LL API cross reference:

- CR1 CRCEN LL_SPI_DisableCRC

LL_SPI_IsEnabledCRC

Function name

__STATIC_INLINE uint32_t LL_SPI_IsEnabledCRC (SPI_TypeDef * SPIx)

Function description

Check if CRC is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Notes

- This bit should be written only when SPI is disabled (SPE = 0) for correct operation.

Reference Manual to LL API cross reference:

- CR1 CRCEN LL_SPI_IsEnabledCRC

LL_SPI_SetCRCNext

Function name

__STATIC_INLINE void LL_SPI_SetCRCNext (SPI_TypeDef * SPIx)

Function description

Set CRCNext to transfer CRC on the line.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- This bit has to be written as soon as the last data is written in the SPIx_DR register.

Reference Manual to LL API cross reference:

- CR1 CRCNEXT LL_SPI_SetCRCNext

LL_SPI_SetCRCPolynomial

Function name

__STATIC_INLINE void LL_SPI_SetCRCPolynomial (SPI_TypeDef * SPIx, uint32_t CRCPoly)

Function description

Set polynomial for CRC calculation.

Parameters

- **SPIx:** SPI Instance
- **CRCPoly:** This parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- CRCPR CRCPOLY LL_SPI_SetCRCPolynomial

LL_SPI_GetCRCPolynomial

Function name

__STATIC_INLINE uint32_t LL_SPI_GetCRCPolynomial (SPI_TypeDef * SPIx)

Function description

Get polynomial for CRC calculation.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value is a number between Min_Data = 0x00 and Max_Data = 0xFFFF

Reference Manual to LL API cross reference:

- CRCPR CRCPOLY LL_SPI_GetCRCPolynomial

LL_SPI_GetRxCRC

Function name

__STATIC_INLINE uint32_t LL_SPI_GetRxCRC (SPI_TypeDef * SPIx)

Function description

Get Rx CRC.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value is a number between Min_Data = 0x00 and Max_Data = 0xFFFF

Reference Manual to LL API cross reference:

- RXCRCR RXCRC LL_SPI_GetRxCRC

LL_SPI_GetTxCRC

Function name

__STATIC_INLINE uint32_t LL_SPI_GetTxCRC (SPI_TypeDef * SPIx)

Function description

Get Tx CRC.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value is a number between Min_Data = 0x00 and Max_Data = 0xFFFF

Reference Manual to LL API cross reference:

- TXCRCR TXCRC LL_SPI_GetTxCRC

LL_SPI_SetNSSMode

Function name

__STATIC_INLINE void LL_SPI_SetNSSMode (SPI_TypeDef * SPIx, uint32_t NSS)

Function description

Set NSS mode.

Parameters

- **SPIx:** SPI Instance
- **NSS:** This parameter can be one of the following values:
  - LL_SPI_NSS_SOFT
  - LL_SPI_NSS_HARD_INPUT
  - LL_SPI_NSS_HARD_OUTPUT

Return values

- **None:**

Notes

- LL_SPI_NSS_SOFT Mode is not used in SPI TI mode.

Reference Manual to LL API cross reference:

- CR1 SSM LL_SPI_SetNSSMode
-
- CR2 SSOE LL_SPI_SetNSSMode

LL_SPI_GetNSSMode

Function name

__STATIC_INLINE uint32_t LL_SPI_GetNSSMode (SPI_TypeDef * SPIx)

Function description

Get NSS mode.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_SPI_NSS_SOFT
  - LL_SPI_NSS_HARD_INPUT
  - LL_SPI_NSS_HARD_OUTPUT

Reference Manual to LL API cross reference:

- CR1 SSM LL_SPI_GetNSSMode
-
- CR2 SSOE LL_SPI_GetNSSMode

LL_SPI_IsActiveFlag_RXNE

Function name

__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_RXNE (SPI_TypeDef * SPIx)

Function description

Check if Rx buffer is not empty.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR RXNE LL_SPI_IsActiveFlag_RXNE

LL_SPI_IsActiveFlag_TXE

Function name

__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_TXE (SPI_TypeDef * SPIx)

Function description

Check if Tx buffer is empty.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR TXE LL_SPI_IsActiveFlag_TXE

LL_SPI_IsActiveFlag_CRCERR

Function name

__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_CRCERR (SPI_TypeDef * SPIx)

Function description

Get CRC error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CRCERR LL_SPI_IsActiveFlag_CRCERR

LL_SPI_IsActiveFlag_MODF

Function name

__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_MODF (SPI_TypeDef * SPIx)

Function description

Get mode fault error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR MODF LL_SPI_IsActiveFlag_MODF

LL_SPI_IsActiveFlag_OVR

Function name

__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_OVR (SPI_TypeDef * SPIx)

Function description

Get overrun error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR OVR LL_SPI_IsActiveFlag_OVR

LL_SPI_IsActiveFlag_BSY

Function name

__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_BSY (SPI_TypeDef * SPIx)

Function description

Get busy flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Notes

- The BSY flag is cleared under any one of the following conditions: -When the SPI is correctly disabled -When a fault is detected in Master mode (MODF bit set to 1) -In Master mode, when it finishes a data transmission and no new data is ready to be sent -In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Reference Manual to LL API cross reference:

- SR BSY LL_SPI_IsActiveFlag_BSY

LL_SPI_IsActiveFlag_FRE

Function name

__STATIC_INLINE uint32_t LL_SPI_IsActiveFlag_FRE (SPI_TypeDef * SPIx)

Function description

Get frame format error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR FRE LL_SPI_IsActiveFlag_FRE

LL_SPI_ClearFlag_CRCERR

Function name

__STATIC_INLINE void LL_SPI_ClearFlag_CRCERR (SPI_TypeDef * SPIx)

Function description

Clear CRC error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CRCERR LL_SPI_ClearFlag_CRCERR

LL_SPI_ClearFlag_MODF

Function name

__STATIC_INLINE void LL_SPI_ClearFlag_MODF (SPI_TypeDef * SPIx)

Function description

Clear mode fault error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the SPIx_SR register followed by a write access to the SPIx_CR1 register

Reference Manual to LL API cross reference:

- SR MODF LL_SPI_ClearFlag_MODF

LL_SPI_ClearFlag_OVR

Function name

__STATIC_INLINE void LL_SPI_ClearFlag_OVR (SPI_TypeDef * SPIx)

Function description

Clear overrun error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the SPIx_DR register followed by a read access to the SPIx_SR register

Reference Manual to LL API cross reference:

- SR OVR LL_SPI_ClearFlag_OVR

LL_SPI_ClearFlag_FRE

Function name

__STATIC_INLINE void LL_SPI_ClearFlag_FRE (SPI_TypeDef * SPIx)

Function description

Clear frame format error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- Clearing this flag is done by reading SPIx_SR register

Reference Manual to LL API cross reference:

- SR FRE LL_SPI_ClearFlag_FRE

LL_SPI_EnableIT_ERR

Function name

__STATIC_INLINE void LL_SPI_EnableIT_ERR (SPI_TypeDef * SPIx)

Function description

Enable error interrupt.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode).

Reference Manual to LL API cross reference:

- CR2 ERRIE LL_SPI_EnableIT_ERR

LL_SPI_EnableIT_RXNE

Function name

__STATIC_INLINE void LL_SPI_EnableIT_RXNE (SPI_TypeDef * SPIx)

Function description

Enable Rx buffer not empty interrupt.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 RXNEIE LL_SPI_EnableIT_RXNE

LL_SPI_EnableIT_TXE

Function name

__STATIC_INLINE void LL_SPI_EnableIT_TXE (SPI_TypeDef * SPIx)

Function description

Enable Tx buffer empty interrupt.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 TXEIE LL_SPI_EnableIT_TXE

LL_SPI_DisableIT_ERR

Function name

__STATIC_INLINE void LL_SPI_DisableIT_ERR (SPI_TypeDef * SPIx)

Function description

Disable error interrupt.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode).

Reference Manual to LL API cross reference:

- CR2 ERRIE LL_SPI_DisableIT_ERR

LL_SPI_DisableIT_RXNE

Function name

__STATIC_INLINE void LL_SPI_DisableIT_RXNE (SPI_TypeDef * SPIx)

Function description

Disable Rx buffer not empty interrupt.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 RXNEIE LL_SPI_DisableIT_RXNE

LL_SPI_DisableIT_TXE

Function name

__STATIC_INLINE void LL_SPI_DisableIT_TXE (SPI_TypeDef * SPIx)

Function description

Disable Tx buffer empty interrupt.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 TXEIE LL_SPI_DisableIT_TXE

LL_SPI_IsEnabledIT_ERR

Function name

__STATIC_INLINE uint32_t LL_SPI_IsEnabledIT_ERR (SPI_TypeDef * SPIx)

Function description

Check if error interrupt is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 ERRIE LL_SPI_IsEnabledIT_ERR

LL_SPI_IsEnabledIT_RXNE

Function name

__STATIC_INLINE uint32_t LL_SPI_IsEnabledIT_RXNE (SPI_TypeDef * SPIx)

Function description

Check if Rx buffer not empty interrupt is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 RXNEIE LL_SPI_IsEnabledIT_RXNE

LL_SPI_IsEnabledIT_TXE

Function name

__STATIC_INLINE uint32_t LL_SPI_IsEnabledIT_TXE (SPI_TypeDef * SPIx)

Function description

Check if Tx buffer empty interrupt.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 TXEIE LL_SPI_IsEnabledIT_TXE

LL_SPI_EnableDMAReq_RX

Function name

__STATIC_INLINE void LL_SPI_EnableDMAReq_RX (SPI_TypeDef * SPIx)

Function description

Enable DMA Rx.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 RXDMAEN LL_SPI_EnableDMAReq_RX

LL_SPI_DisableDMAReq_RX

Function name

__STATIC_INLINE void LL_SPI_DisableDMAReq_RX (SPI_TypeDef * SPIx)

Function description

Disable DMA Rx.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 RXDMAEN LL_SPI_DisableDMAReq_RX

LL_SPI_IsEnabledDMAReq_RX

Function name

__STATIC_INLINE uint32_t LL_SPI_IsEnabledDMAReq_RX (SPI_TypeDef * SPIx)

Function description

Check if DMA Rx is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 RXDMAEN LL_SPI_IsEnabledDMAReq_RX

LL_SPI_EnableDMAReq_TX

Function name

__STATIC_INLINE void LL_SPI_EnableDMAReq_TX (SPI_TypeDef * SPIx)

Function description

Enable DMA Tx.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 TXDMAEN LL_SPI_EnableDMAReq_TX

LL_SPI_DisableDMAReq_TX

Function name

__STATIC_INLINE void LL_SPI_DisableDMAReq_TX (SPI_TypeDef * SPIx)

Function description

Disable DMA Tx.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 TXDMAEN LL_SPI_DisableDMAReq_TX

LL_SPI_IsEnabledDMAReq_TX

Function name

__STATIC_INLINE uint32_t LL_SPI_IsEnabledDMAReq_TX (SPI_TypeDef * SPIx)

Function description

Check if DMA Tx is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 TXDMAEN LL_SPI_IsEnabledDMAReq_TX

LL_SPI_DMA_GetRegAddr

Function name

__STATIC_INLINE uint32_t LL_SPI_DMA_GetRegAddr (SPI_TypeDef * SPIx)

Function description

Get the data register address used for DMA transfer.

Parameters

- **SPIx:** SPI Instance

Return values

- **Address:** of data register

Reference Manual to LL API cross reference:

- DR DR LL_SPI_DMA_GetRegAddr

LL_SPI_ReceiveData8

Function name

__STATIC_INLINE uint8_t LL_SPI_ReceiveData8 (SPI_TypeDef * SPIx)

Function description

Read 8-Bits in the data register.

Parameters

- **SPIx:** SPI Instance

Return values

- **RxData:** Value between Min_Data=0x00 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- DR DR LL_SPI_ReceiveData8

LL_SPI_ReceiveData16

Function name

__STATIC_INLINE uint16_t LL_SPI_ReceiveData16 (SPI_TypeDef * SPIx)

Function description

Read 16-Bits in the data register.

Parameters

- **SPIx:** SPI Instance

Return values

- **RxData:** Value between Min_Data=0x00 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- DR DR LL_SPI_ReceiveData16

LL_SPI_TransmitData8

Function name

__STATIC_INLINE void LL_SPI_TransmitData8 (SPI_TypeDef * SPIx, uint8_t TxData)

Function description

Write 8-Bits in the data register.

Parameters

- **SPIx:** SPI Instance
- **TxData:** Value between Min_Data=0x00 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR DR LL_SPI_TransmitData8

LL_SPI_TransmitData16

Function name

__STATIC_INLINE void LL_SPI_TransmitData16 (SPI_TypeDef * SPIx, uint16_t TxData)

Function description

Write 16-Bits in the data register.

Parameters

- **SPIx:** SPI Instance
- **TxData:** Value between Min_Data=0x00 and Max_Data=0xFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR DR LL_SPI_TransmitData16

LL_SPI_DeInit

Function name

ErrorStatus LL_SPI_DeInit (SPI_TypeDef * SPIx)

Function description

De-initialize the SPI registers to their default reset values.

Parameters

- **SPIx:** SPI Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: SPI registers are de-initialized
  - ERROR: SPI registers are not de-initialized

LL_SPI_Init

Function name

ErrorStatus LL_SPI_Init (SPI_TypeDef * SPIx, LL_SPI_InitTypeDef * SPI_InitStruct)

Function description

Initialize the SPI registers according to the specified parameters in SPI_InitStruct.

Parameters

- **SPIx:** SPI Instance
- **SPI_InitStruct:** pointer to a LL_SPI_InitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value. (Return always SUCCESS)

Notes

- As some bits in SPI configuration registers can only be written when the SPI is disabled (SPI_CR1_SPE bit =0), SPI peripheral should be in disabled state prior calling this function. Otherwise, ERROR result will be returned.

LL_SPI_StructInit

Function name

void LL_SPI_StructInit (LL_SPI_InitTypeDef * SPI_InitStruct)

Function description

Set each LL_SPI_InitTypeDef field to default value.

Parameters

- **SPI_InitStruct:** pointer to a LL_SPI_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

LL_I2S_Enable

Function name

__STATIC_INLINE void LL_I2S_Enable (SPI_TypeDef * SPIx)

Function description

Select I2S mode and Enable I2S peripheral.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SCFGR I2SMOD LL_I2S_Enable
- I2SCFGR I2SE LL_I2S_Enable

LL_I2S_Disable

Function name

__STATIC_INLINE void LL_I2S_Disable (SPI_TypeDef * SPIx)

Function description

Disable I2S peripheral.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SCFGR I2SE LL_I2S_Disable

LL_I2S_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_I2S_IsEnabled (SPI_TypeDef * SPIx)

Function description

Check if I2S peripheral is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- I2SCFGR I2SE LL_I2S_IsEnabled

LL_I2S_SetDataFormat

Function name

__STATIC_INLINE void LL_I2S_SetDataFormat (SPI_TypeDef * SPIx, uint32_t DataFormat)

Function description

Set I2S data frame length.

Parameters

- **SPIx:** SPI Instance
- **DataFormat:** This parameter can be one of the following values:
  - LL_I2S_DATAFORMAT_16B
  - LL_I2S_DATAFORMAT_16B_EXTENDED
  - LL_I2S_DATAFORMAT_24B
  - LL_I2S_DATAFORMAT_32B

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SCFGR DATLEN LL_I2S_SetDataFormat
- I2SCFGR CHLEN LL_I2S_SetDataFormat

LL_I2S_GetDataFormat

Function name

__STATIC_INLINE uint32_t LL_I2S_GetDataFormat (SPI_TypeDef * SPIx)

Function description

Get I2S data frame length.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_I2S_DATAFORMAT_16B
  - LL_I2S_DATAFORMAT_16B_EXTENDED
  - LL_I2S_DATAFORMAT_24B
  - LL_I2S_DATAFORMAT_32B

Reference Manual to LL API cross reference:

- I2SCFGR DATLEN LL_I2S_GetDataFormat
- I2SCFGR CHLEN LL_I2S_GetDataFormat

LL_I2S_SetClockPolarity

Function name

__STATIC_INLINE void LL_I2S_SetClockPolarity (SPI_TypeDef * SPIx, uint32_t ClockPolarity)

Function description

Set I2S clock polarity.

Parameters

- **SPIx:** SPI Instance
- **ClockPolarity:** This parameter can be one of the following values:
  - LL_I2S_POLARITY_LOW
  - LL_I2S_POLARITY_HIGH

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SCFGR CKPOL LL_I2S_SetClockPolarity

LL_I2S_GetClockPolarity

Function name

__STATIC_INLINE uint32_t LL_I2S_GetClockPolarity (SPI_TypeDef * SPIx)

Function description

Get I2S clock polarity.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_I2S_POLARITY_LOW
  - LL_I2S_POLARITY_HIGH

Reference Manual to LL API cross reference:

- I2SCFGR CKPOL LL_I2S_GetClockPolarity

LL_I2S_SetStandard

Function name

__STATIC_INLINE void LL_I2S_SetStandard (SPI_TypeDef * SPIx, uint32_t Standard)

Function description

Set I2S standard protocol.

Parameters

- **SPIx:** SPI Instance
- **Standard:** This parameter can be one of the following values:
  - LL_I2S_STANDARD_PHILIPS
  - LL_I2S_STANDARD_MSB
  - LL_I2S_STANDARD_LSB
  - LL_I2S_STANDARD_PCM_SHORT
  - LL_I2S_STANDARD_PCM_LONG

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SCFGR I2SSTD LL_I2S_SetStandard
- I2SCFGR PCMSYNC LL_I2S_SetStandard

LL_I2S_GetStandard

Function name

__STATIC_INLINE uint32_t LL_I2S_GetStandard (SPI_TypeDef * SPIx)

Function description

Get I2S standard protocol.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_I2S_STANDARD_PHILIPS
  - LL_I2S_STANDARD_MSB
  - LL_I2S_STANDARD_LSB
  - LL_I2S_STANDARD_PCM_SHORT
  - LL_I2S_STANDARD_PCM_LONG

Reference Manual to LL API cross reference:

- I2SCFGR I2SSTD LL_I2S_GetStandard
- I2SCFGR PCMSYNC LL_I2S_GetStandard

LL_I2S_SetTransferMode

Function name

__STATIC_INLINE void LL_I2S_SetTransferMode (SPI_TypeDef * SPIx, uint32_t Mode)

Function description

Set I2S transfer mode.

Parameters

- **SPIx:** SPI Instance
- **Mode:** This parameter can be one of the following values:
  - LL_I2S_MODE_SLAVE_TX
  - LL_I2S_MODE_SLAVE_RX
  - LL_I2S_MODE_MASTER_TX
  - LL_I2S_MODE_MASTER_RX

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SCFGR I2SCFG LL_I2S_SetTransferMode

LL_I2S_GetTransferMode

Function name

__STATIC_INLINE uint32_t LL_I2S_GetTransferMode (SPI_TypeDef * SPIx)

Function description

Get I2S transfer mode.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_I2S_MODE_SLAVE_TX
  - LL_I2S_MODE_SLAVE_RX
  - LL_I2S_MODE_MASTER_TX
  - LL_I2S_MODE_MASTER_RX

Reference Manual to LL API cross reference:

- I2SCFGR I2SCFG LL_I2S_GetTransferMode

LL_I2S_SetPrescalerLinear

Function name

__STATIC_INLINE void LL_I2S_SetPrescalerLinear (SPI_TypeDef * SPIx, uint8_t PrescalerLinear)

Function description

Set I2S linear prescaler.

Parameters

- **SPIx:** SPI Instance
- **PrescalerLinear:** Value between Min_Data=0x02 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SPR I2SDIV LL_I2S_SetPrescalerLinear

LL_I2S_GetPrescalerLinear

Function name

__STATIC_INLINE uint32_t LL_I2S_GetPrescalerLinear (SPI_TypeDef * SPIx)

Function description

Get I2S linear prescaler.

Parameters

- **SPIx:** SPI Instance

Return values

- **PrescalerLinear:** Value between Min_Data=0x02 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- I2SPR I2SDIV LL_I2S_GetPrescalerLinear

LL_I2S_SetPrescalerParity

Function name

__STATIC_INLINE void LL_I2S_SetPrescalerParity (SPI_TypeDef * SPIx, uint32_t PrescalerParity)

Function description

Set I2S parity prescaler.

Parameters

- **SPIx:** SPI Instance
- **PrescalerParity:** This parameter can be one of the following values:
  - LL_I2S_PRESCALER_PARITY_EVEN
  - LL_I2S_PRESCALER_PARITY_ODD

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SPR ODD LL_I2S_SetPrescalerParity

LL_I2S_GetPrescalerParity

Function name

__STATIC_INLINE uint32_t LL_I2S_GetPrescalerParity (SPI_TypeDef * SPIx)

Function description

Get I2S parity prescaler.

Parameters

- **SPIx:** SPI Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_I2S_PRESCALER_PARITY_EVEN
  - LL_I2S_PRESCALER_PARITY_ODD

Reference Manual to LL API cross reference:

- I2SPR ODD LL_I2S_GetPrescalerParity

LL_I2S_EnableMasterClock

Function name

__STATIC_INLINE void LL_I2S_EnableMasterClock (SPI_TypeDef * SPIx)

Function description

Enable the master clock ouput (Pin MCK)

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SPR MCKOE LL_I2S_EnableMasterClock

LL_I2S_DisableMasterClock

Function name

__STATIC_INLINE void LL_I2S_DisableMasterClock (SPI_TypeDef * SPIx)

Function description

Disable the master clock ouput (Pin MCK)

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SPR MCKOE LL_I2S_DisableMasterClock

LL_I2S_IsEnabledMasterClock

Function name

__STATIC_INLINE uint32_t LL_I2S_IsEnabledMasterClock (SPI_TypeDef * SPIx)

Function description

Check if the master clock ouput (Pin MCK) is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- I2SPR MCKOE LL_I2S_IsEnabledMasterClock

LL_I2S_EnableAsyncStart

Function name

__STATIC_INLINE void LL_I2S_EnableAsyncStart (SPI_TypeDef * SPIx)

Function description

Enable asynchronous start.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SCFGR ASTRTEN LL_I2S_EnableAsyncStart

LL_I2S_DisableAsyncStart

Function name

__STATIC_INLINE void LL_I2S_DisableAsyncStart (SPI_TypeDef * SPIx)

Function description

Disable asynchronous start.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- I2SCFGR ASTRTEN LL_I2S_DisableAsyncStart

LL_I2S_IsEnabledAsyncStart

Function name

__STATIC_INLINE uint32_t LL_I2S_IsEnabledAsyncStart (SPI_TypeDef * SPIx)

Function description

Check if asynchronous start is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- I2SCFGR ASTRTEN LL_I2S_IsEnabledAsyncStart

LL_I2S_IsActiveFlag_RXNE

Function name

__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_RXNE (SPI_TypeDef * SPIx)

Function description

Check if Rx buffer is not empty.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR RXNE LL_I2S_IsActiveFlag_RXNE

LL_I2S_IsActiveFlag_TXE

Function name

__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_TXE (SPI_TypeDef * SPIx)

Function description

Check if Tx buffer is empty.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR TXE LL_I2S_IsActiveFlag_TXE

LL_I2S_IsActiveFlag_BSY

Function name

__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_BSY (SPI_TypeDef * SPIx)

Function description

Get busy flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR BSY LL_I2S_IsActiveFlag_BSY

LL_I2S_IsActiveFlag_OVR

Function name

__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_OVR (SPI_TypeDef * SPIx)

Function description

Get overrun error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR OVR LL_I2S_IsActiveFlag_OVR

LL_I2S_IsActiveFlag_UDR

Function name

__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_UDR (SPI_TypeDef * SPIx)

Function description

Get underrun error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR UDR LL_I2S_IsActiveFlag_UDR

LL_I2S_IsActiveFlag_FRE

Function name

__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_FRE (SPI_TypeDef * SPIx)

Function description

Get frame format error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR FRE LL_I2S_IsActiveFlag_FRE

LL_I2S_IsActiveFlag_CHSIDE

Function name

__STATIC_INLINE uint32_t LL_I2S_IsActiveFlag_CHSIDE (SPI_TypeDef * SPIx)

Function description

Get channel side flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Notes

- 0: Channel Left has to be transmitted or has been received 1: Channel Right has to be transmitted or has been received It has no significance in PCM mode.

Reference Manual to LL API cross reference:

- SR CHSIDE LL_I2S_IsActiveFlag_CHSIDE

LL_I2S_ClearFlag_OVR

Function name

__STATIC_INLINE void LL_I2S_ClearFlag_OVR (SPI_TypeDef * SPIx)

Function description

Clear overrun error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR OVR LL_I2S_ClearFlag_OVR

LL_I2S_ClearFlag_UDR

Function name

__STATIC_INLINE void LL_I2S_ClearFlag_UDR (SPI_TypeDef * SPIx)

Function description

Clear underrun error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR UDR LL_I2S_ClearFlag_UDR

LL_I2S_ClearFlag_FRE

Function name

__STATIC_INLINE void LL_I2S_ClearFlag_FRE (SPI_TypeDef * SPIx)

Function description

Clear frame format error flag.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR FRE LL_I2S_ClearFlag_FRE

LL_I2S_EnableIT_ERR

Function name

__STATIC_INLINE void LL_I2S_EnableIT_ERR (SPI_TypeDef * SPIx)

Function description

Enable error IT.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- This bit controls the generation of an interrupt when an error condition occurs (OVR, UDR and FRE in I2S mode).

Reference Manual to LL API cross reference:

- CR2 ERRIE LL_I2S_EnableIT_ERR

LL_I2S_EnableIT_RXNE

Function name

__STATIC_INLINE void LL_I2S_EnableIT_RXNE (SPI_TypeDef * SPIx)

Function description

Enable Rx buffer not empty IT.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 RXNEIE LL_I2S_EnableIT_RXNE

LL_I2S_EnableIT_TXE

Function name

__STATIC_INLINE void LL_I2S_EnableIT_TXE (SPI_TypeDef * SPIx)

Function description

Enable Tx buffer empty IT.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 TXEIE LL_I2S_EnableIT_TXE

LL_I2S_DisableIT_ERR

Function name

__STATIC_INLINE void LL_I2S_DisableIT_ERR (SPI_TypeDef * SPIx)

Function description

Disable error IT.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Notes

- This bit controls the generation of an interrupt when an error condition occurs (OVR, UDR and FRE in I2S mode).

Reference Manual to LL API cross reference:

- CR2 ERRIE LL_I2S_DisableIT_ERR

LL_I2S_DisableIT_RXNE

Function name

__STATIC_INLINE void LL_I2S_DisableIT_RXNE (SPI_TypeDef * SPIx)

Function description

Disable Rx buffer not empty IT.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 RXNEIE LL_I2S_DisableIT_RXNE

LL_I2S_DisableIT_TXE

Function name

__STATIC_INLINE void LL_I2S_DisableIT_TXE (SPI_TypeDef * SPIx)

Function description

Disable Tx buffer empty IT.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 TXEIE LL_I2S_DisableIT_TXE

LL_I2S_IsEnabledIT_ERR

Function name

__STATIC_INLINE uint32_t LL_I2S_IsEnabledIT_ERR (SPI_TypeDef * SPIx)

Function description

Check if ERR IT is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 ERRIE LL_I2S_IsEnabledIT_ERR

LL_I2S_IsEnabledIT_RXNE

Function name

__STATIC_INLINE uint32_t LL_I2S_IsEnabledIT_RXNE (SPI_TypeDef * SPIx)

Function description

Check if RXNE IT is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 RXNEIE LL_I2S_IsEnabledIT_RXNE

LL_I2S_IsEnabledIT_TXE

Function name

__STATIC_INLINE uint32_t LL_I2S_IsEnabledIT_TXE (SPI_TypeDef * SPIx)

Function description

Check if TXE IT is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 TXEIE LL_I2S_IsEnabledIT_TXE

LL_I2S_EnableDMAReq_RX

Function name

__STATIC_INLINE void LL_I2S_EnableDMAReq_RX (SPI_TypeDef * SPIx)

Function description

Enable DMA Rx.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 RXDMAEN LL_I2S_EnableDMAReq_RX

LL_I2S_DisableDMAReq_RX

Function name

__STATIC_INLINE void LL_I2S_DisableDMAReq_RX (SPI_TypeDef * SPIx)

Function description

Disable DMA Rx.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 RXDMAEN LL_I2S_DisableDMAReq_RX

LL_I2S_IsEnabledDMAReq_RX

Function name

__STATIC_INLINE uint32_t LL_I2S_IsEnabledDMAReq_RX (SPI_TypeDef * SPIx)

Function description

Check if DMA Rx is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 RXDMAEN LL_I2S_IsEnabledDMAReq_RX

LL_I2S_EnableDMAReq_TX

Function name

__STATIC_INLINE void LL_I2S_EnableDMAReq_TX (SPI_TypeDef * SPIx)

Function description

Enable DMA Tx.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 TXDMAEN LL_I2S_EnableDMAReq_TX

LL_I2S_DisableDMAReq_TX

Function name

__STATIC_INLINE void LL_I2S_DisableDMAReq_TX (SPI_TypeDef * SPIx)

Function description

Disable DMA Tx.

Parameters

- **SPIx:** SPI Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 TXDMAEN LL_I2S_DisableDMAReq_TX

LL_I2S_IsEnabledDMAReq_TX

Function name

__STATIC_INLINE uint32_t LL_I2S_IsEnabledDMAReq_TX (SPI_TypeDef * SPIx)

Function description

Check if DMA Tx is enabled.

Parameters

- **SPIx:** SPI Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR2 TXDMAEN LL_I2S_IsEnabledDMAReq_TX

LL_I2S_ReceiveData16

Function name

__STATIC_INLINE uint16_t LL_I2S_ReceiveData16 (SPI_TypeDef * SPIx)

Function description

Read 16-Bits in data register.

Parameters

- **SPIx:** SPI Instance

Return values

- **RxData:** Value between Min_Data=0x0000 and Max_Data=0xFFFF

Reference Manual to LL API cross reference:

- DR DR LL_I2S_ReceiveData16

LL_I2S_TransmitData16

Function name

__STATIC_INLINE void LL_I2S_TransmitData16 (SPI_TypeDef * SPIx, uint16_t TxData)

Function description

Write 16-Bits in data register.

Parameters

- **SPIx:** SPI Instance
- **TxData:** Value between Min_Data=0x0000 and Max_Data=0xFFFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR DR LL_I2S_TransmitData16

LL_I2S_DeInit

Function name

ErrorStatus LL_I2S_DeInit (SPI_TypeDef * SPIx)

Function description

De-initialize the SPI/I2S registers to their default reset values.

Parameters

- **SPIx:** SPI Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: SPI registers are de-initialized
  - ERROR: SPI registers are not de-initialized

LL_I2S_Init

Function name

ErrorStatus LL_I2S_Init (SPI_TypeDef * SPIx, LL_I2S_InitTypeDef * I2S_InitStruct)

Function description

Initializes the SPI/I2S registers according to the specified parameters in I2S_InitStruct.

Parameters

- **SPIx:** SPI Instance
- **I2S_InitStruct:** pointer to a LL_I2S_InitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: SPI registers are Initialized
  - ERROR: SPI registers are not Initialized

Notes

- As some bits in SPI configuration registers can only be written when the SPI is disabled (SPI_CR1_SPE bit =0), SPI peripheral should be in disabled state prior calling this function. Otherwise, ERROR result will be returned.

LL_I2S_StructInit

Function name

void LL_I2S_StructInit (LL_I2S_InitTypeDef * I2S_InitStruct)

Function description

Set each LL_I2S_InitTypeDef field to default value.

Parameters

- **I2S_InitStruct:** pointer to a LL_I2S_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

LL_I2S_ConfigPrescaler

Function name

void LL_I2S_ConfigPrescaler (SPI_TypeDef * SPIx, uint32_t PrescalerLinear, uint32_t PrescalerParity)

Function description

Set linear and parity prescaler.

Parameters

- **SPIx:** SPI Instance
- **PrescalerLinear:** value Min_Data=0x02 and Max_Data=0xFF.
- **PrescalerParity:** This parameter can be one of the following values:
  - LL_I2S_PRESCALER_PARITY_EVEN
  - LL_I2S_PRESCALER_PARITY_ODD

Return values

- **None:**

Notes

- To calculate value of PrescalerLinear(I2SDIV[7:0] bits) and PrescalerParity(ODD bit) Check Audio frequency table and formulas inside Reference Manual (SPI/I2S).

LL_I2S_InitFullDuplex

Function name

ErrorStatus LL_I2S_InitFullDuplex (SPI_TypeDef * I2Sxext, LL_I2S_InitTypeDef * I2S_InitStruct)

Function description

Configures the full duplex mode for the I2Sx peripheral using its extension I2Sxext according to the specified parameters in the I2S_InitStruct.

Parameters

- **I2Sxext:** SPI Instance
- **I2S_InitStruct:** pointer to a LL_I2S_InitTypeDef structure

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: I2Sxext registers are Initialized
  - ERROR: I2Sxext registers are not Initialized

Notes

- The structure pointed by I2S_InitStruct parameter should be the same used for the master I2S peripheral. In this case, if the master is configured as transmitter, the slave will be receiver and vice versa. Or you can force a different mode by modifying the field I2S_Mode to the value I2S_SlaveRx or I2S_SlaveTx independently of the master configuration.

90.3 SPI Firmware driver defines

The following section lists the various define and macros of the module.

90.3.1 SPI

SPI

Baud Rate Prescaler

LL_SPI_BAUDRATEPRESCALER_DIV2

BaudRate control equal to fPCLK/2

LL_SPI_BAUDRATEPRESCALER_DIV4

BaudRate control equal to fPCLK/4

LL_SPI_BAUDRATEPRESCALER_DIV8

BaudRate control equal to fPCLK/8

LL_SPI_BAUDRATEPRESCALER_DIV16

BaudRate control equal to fPCLK/16

LL_SPI_BAUDRATEPRESCALER_DIV32

BaudRate control equal to fPCLK/32

LL_SPI_BAUDRATEPRESCALER_DIV64

BaudRate control equal to fPCLK/64

LL_SPI_BAUDRATEPRESCALER_DIV128

BaudRate control equal to fPCLK/128

LL_SPI_BAUDRATEPRESCALER_DIV256

BaudRate control equal to fPCLK/256

Transmission Bit Order

LL_SPI_LSB_FIRST

Data is transmitted/received with the LSB first

LL_SPI_MSB_FIRST

Data is transmitted/received with the MSB first

CRC Calculation

LL_SPI_CRCCALCULATION_DISABLE

CRC calculation disabled

LL_SPI_CRCCALCULATION_ENABLE

CRC calculation enabled

Datawidth

LL_SPI_DATAWIDTH_8BIT

Data length for SPI transfer: 8 bits

LL_SPI_DATAWIDTH_16BIT

Data length for SPI transfer: 16 bits

Get Flags Defines

LL_SPI_SR_RXNE

Rx buffer not empty flag

LL_SPI_SR_TXE

Tx buffer empty flag

LL_SPI_SR_BSY

Busy flag

LL_SPI_SR_CRCERR

CRC error flag

LL_SPI_SR_MODF

Mode fault flag

LL_SPI_SR_OVR

Overrun flag

LL_SPI_SR_FRE

TI mode frame format error flag

IT Defines

LL_SPI_CR2_RXNEIE

Rx buffer not empty interrupt enable

LL_SPI_CR2_TXEIE

Tx buffer empty interrupt enable

LL_SPI_CR2_ERRIE

Error interrupt enable

LL_I2S_CR2_RXNEIE

Rx buffer not empty interrupt enable

LL_I2S_CR2_TXEIE

Tx buffer empty interrupt enable

LL_I2S_CR2_ERRIE

Error interrupt enable

Operation Mode

LL_SPI_MODE_MASTER

Master configuration

LL_SPI_MODE_SLAVE

Slave configuration

Slave Select Pin Mode

LL_SPI_NSS_SOFT

NSS managed internally. NSS pin not used and free

LL_SPI_NSS_HARD_INPUT

NSS pin used in Input. Only used in Master mode

LL_SPI_NSS_HARD_OUTPUT

NSS pin used in Output. Only used in Slave mode as chip select

Clock Phase

LL_SPI_PHASE_1EDGE

First clock transition is the first data capture edge

LL_SPI_PHASE_2EDGE

Second clock transition is the first data capture edge

Clock Polarity

LL_SPI_POLARITY_LOW

Clock to 0 when idle

LL_SPI_POLARITY_HIGH

Clock to 1 when idle

Serial Protocol

LL_SPI_PROTOCOL_MOTOROLA

Motorola mode. Used as default value

LL_SPI_PROTOCOL_TI

TI mode

Transfer Mode

LL_SPI_FULL_DUPLEX

Full-Duplex mode. Rx and Tx transfer on 2 lines

LL_SPI_SIMPLEX_RX

Simplex Rx mode. Rx transfer only on 1 line

LL_SPI_HALF_DUPLEX_RX

Half-Duplex Rx mode. Rx transfer on 1 line

LL_SPI_HALF_DUPLEX_TX

Half-Duplex Tx mode. Tx transfer on 1 line

Common Write and read registers Macros

LL_SPI_WriteReg

Description:

- Write a value in SPI register.

Parameters:

- __INSTANCE__: SPI Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_SPI_ReadReg

Description:

- Read a value in SPI register.

Parameters:

- __INSTANCE__: SPI Instance
- __REG__: Register to be read

Return value:

- Register: value

91 LL SYSTEM Generic Driver

91.1 SYSTEM Firmware driver API description

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

91.1.1 Detailed description of functions

LL_SYSCFG_SetRemapMemory

Function name

__STATIC_INLINE void LL_SYSCFG_SetRemapMemory (uint32_t Memory)

Function description

Set memory mapping at address 0x00000000.

Parameters

- **Memory:** This parameter can be one of the following values:
  - LL_SYSCFG_REMAP_FLASH
  - LL_SYSCFG_REMAP_SYSTEMFLASH
  - LL_SYSCFG_REMAP_SRAM
  - LL_SYSCFG_REMAP_FSMC (*)
  - LL_SYSCFG_REMAP_FMC (*)

Return values

- **None:**

Reference Manual to LL API cross reference:

- SYSCFG_MEMRMP MEM_MODE LL_SYSCFG_SetRemapMemory

LL_SYSCFG_GetRemapMemory

Function name

__STATIC_INLINE uint32_t LL_SYSCFG_GetRemapMemory (void)

Function description

Get memory mapping at address 0x00000000.

Return values

- **Returned:** value can be one of the following values:
  - LL_SYSCFG_REMAP_FLASH
  - LL_SYSCFG_REMAP_SYSTEMFLASH
  - LL_SYSCFG_REMAP_SRAM
  - LL_SYSCFG_REMAP_FSMC (*)
  - LL_SYSCFG_REMAP_FMC (*)

Reference Manual to LL API cross reference:

- SYSCFG_MEMRMP MEM_MODE LL_SYSCFG_GetRemapMemory

LL_SYSCFG_EnableFMCMemorySwapping

Function name

__STATIC_INLINE void LL_SYSCFG_EnableFMCMemorySwapping (void)

Function description

Enables the FMC Memory Mapping Swapping.

Return values

- **None:**

Notes

- SDRAM is accessible at 0x60000000 and NOR/RAM is accessible at 0xC0000000

Reference Manual to LL API cross reference:

- SYSCFG_MEMRMP SWP_FMC LL_SYSCFG_EnableFMCMemorySwapping

LL_SYSCFG_DisableFMCMemorySwapping

Function name

__STATIC_INLINE void LL_SYSCFG_DisableFMCMemorySwapping (void)

Function description

Disables the FMC Memory Mapping Swapping.

Return values

- **None:**

Notes

- SDRAM is accessible at 0xC0000000 (default mapping) and NOR/RAM is accessible at 0x60000000 (default mapping)

Reference Manual to LL API cross reference:

- SYSCFG_MEMRMP SWP_FMC LL_SYSCFG_DisableFMCMemorySwapping

LL_SYSCFG_EnableCompensationCell

Function name

__STATIC_INLINE void LL_SYSCFG_EnableCompensationCell (void)

Function description

Enables the Compensation cell Power Down.

Return values

- **None:**

Notes

- The I/O compensation cell can be used only when the device supply voltage ranges from 2.4 to 3.6 V

Reference Manual to LL API cross reference:

- SYSCFG_CMPCR CMP_PD LL_SYSCFG_EnableCompensationCell

LL_SYSCFG_DisableCompensationCell

Function name

__STATIC_INLINE void LL_SYSCFG_DisableCompensationCell (void)

Function description

Disables the Compensation cell Power Down.

Return values

- **None:**

Notes

- The I/O compensation cell can be used only when the device supply voltage ranges from 2.4 to 3.6 V

Reference Manual to LL API cross reference:

- SYSCFG_CMPCR CMP_PD LL_SYSCFG_DisableCompensationCell

LL_SYSCFG_IsActiveFlag_CMPCR

Function name

__STATIC_INLINE uint32_t LL_SYSCFG_IsActiveFlag_CMPCR (void)

Function description

Get Compensation Cell ready Flag.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SYSCFG_CMPCR READY LL_SYSCFG_IsActiveFlag_CMPCR

LL_SYSCFG_SetPHYInterface

Function name

__STATIC_INLINE void LL_SYSCFG_SetPHYInterface (uint32_t Interface)

Function description

Select Ethernet PHY interface.

Parameters

- **Interface:** This parameter can be one of the following values:
  - LL_SYSCFG_PMC_ETHMII
  - LL_SYSCFG_PMC_ETHRMII

Return values

- **None:**

Reference Manual to LL API cross reference:

- SYSCFG_PMC MII_RMII_SEL LL_SYSCFG_SetPHYInterface

LL_SYSCFG_GetPHYInterface

Function name

__STATIC_INLINE uint32_t LL_SYSCFG_GetPHYInterface (void)

Function description

Get Ethernet PHY interface.

Return values

- **Returned:** value can be one of the following values:
  - LL_SYSCFG_PMC_ETHMII
  - LL_SYSCFG_PMC_ETHRMII
- **None:**

Reference Manual to LL API cross reference:

- SYSCFG_PMC MII_RMII_SEL LL_SYSCFG_GetPHYInterface

LL_SYSCFG_SetFlashBankMode

Function name

__STATIC_INLINE void LL_SYSCFG_SetFlashBankMode (uint32_t Bank)

Function description

Select Flash bank mode (Bank flashed at 0x08000000)

Parameters

- **Bank:** This parameter can be one of the following values:
  - LL_SYSCFG_BANKMODE_BANK1
  - LL_SYSCFG_BANKMODE_BANK2

Return values

- **None:**

Reference Manual to LL API cross reference:

- SYSCFG_MEMRMP UFB_MODE LL_SYSCFG_SetFlashBankMode

LL_SYSCFG_GetFlashBankMode

Function name

__STATIC_INLINE uint32_t LL_SYSCFG_GetFlashBankMode (void)

Function description

Get Flash bank mode (Bank flashed at 0x08000000)

Return values

- **Returned:** value can be one of the following values:
  - LL_SYSCFG_BANKMODE_BANK1
  - LL_SYSCFG_BANKMODE_BANK2

Reference Manual to LL API cross reference:

- SYSCFG_MEMRMP UFB_MODE LL_SYSCFG_GetFlashBankMode

LL_SYSCFG_SetEXTISource

Function name

__STATIC_INLINE void LL_SYSCFG_SetEXTISource (uint32_t Port, uint32_t Line)

Function description

Configure source input for the EXTI external interrupt.

Parameters

- **Port:** This parameter can be one of the following values:
  - LL_SYSCFG_EXTI_PORTA
  - LL_SYSCFG_EXTI_PORTB
  - LL_SYSCFG_EXTI_PORTC
  - LL_SYSCFG_EXTI_PORTD
  - LL_SYSCFG_EXTI_PORTE
  - LL_SYSCFG_EXTI_PORTF (*)
  - LL_SYSCFG_EXTI_PORTG (*)
  - LL_SYSCFG_EXTI_PORTH

  (*) value not defined in all devices
- **Line:** This parameter can be one of the following values:
  - LL_SYSCFG_EXTI_LINE0
  - LL_SYSCFG_EXTI_LINE1
  - LL_SYSCFG_EXTI_LINE2
  - LL_SYSCFG_EXTI_LINE3
  - LL_SYSCFG_EXTI_LINE4
  - LL_SYSCFG_EXTI_LINE5
  - LL_SYSCFG_EXTI_LINE6
  - LL_SYSCFG_EXTI_LINE7
  - LL_SYSCFG_EXTI_LINE8
  - LL_SYSCFG_EXTI_LINE9
  - LL_SYSCFG_EXTI_LINE10
  - LL_SYSCFG_EXTI_LINE11
  - LL_SYSCFG_EXTI_LINE12
  - LL_SYSCFG_EXTI_LINE13
  - LL_SYSCFG_EXTI_LINE14
  - LL_SYSCFG_EXTI_LINE15

Return values

- **None:**

Reference Manual to LL API cross reference:

- SYSCFG_EXTICR1 EXTIx LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR2 EXTIx LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR3 EXTIx LL_SYSCFG_SetEXTISource
- SYSCFG_EXTICR4 EXTIx LL_SYSCFG_SetEXTISource

LL_SYSCFG_GetEXTISource

Function name

__STATIC_INLINE uint32_t LL_SYSCFG_GetEXTISource (uint32_t Line)

Function description

Get the configured defined for specific EXTI Line.

Parameters

- **Line:** This parameter can be one of the following values:
  - LL_SYSCFG_EXTI_LINE0
  - LL_SYSCFG_EXTI_LINE1
  - LL_SYSCFG_EXTI_LINE2
  - LL_SYSCFG_EXTI_LINE3
  - LL_SYSCFG_EXTI_LINE4
  - LL_SYSCFG_EXTI_LINE5
  - LL_SYSCFG_EXTI_LINE6
  - LL_SYSCFG_EXTI_LINE7
  - LL_SYSCFG_EXTI_LINE8
  - LL_SYSCFG_EXTI_LINE9
  - LL_SYSCFG_EXTI_LINE10
  - LL_SYSCFG_EXTI_LINE11
  - LL_SYSCFG_EXTI_LINE12
  - LL_SYSCFG_EXTI_LINE13
  - LL_SYSCFG_EXTI_LINE14
  - LL_SYSCFG_EXTI_LINE15

Return values

- **Returned:** value can be one of the following values:
  - LL_SYSCFG_EXTI_PORTA
  - LL_SYSCFG_EXTI_PORTB
  - LL_SYSCFG_EXTI_PORTC
  - LL_SYSCFG_EXTI_PORTD
  - LL_SYSCFG_EXTI_PORTE
  - LL_SYSCFG_EXTI_PORTF (*)
  - LL_SYSCFG_EXTI_PORTG (*)
  - LL_SYSCFG_EXTI_PORTH (*) value not defined in all devices

Reference Manual to LL API cross reference:

- SYSCFG_EXTICR1 EXTIx LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR2 EXTIx LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR3 EXTIx LL_SYSCFG_GetEXTISource
- SYSCFG_EXTICR4 EXTIx LL_SYSCFG_GetEXTISource

LL_DBGMCU_GetDeviceID

Function name

__STATIC_INLINE uint32_t LL_DBGMCU_GetDeviceID (void)

Function description

Return the device identifier.

Return values

- **Values:** between Min_Data=0x00 and Max_Data=0xFFF

Notes

- For STM32F405/407xx and STM32F415/417xx devices, the device ID is 0x413
- For STM32F42xxx and STM32F43xxx devices, the device ID is 0x419
- For STM32F401xx devices, the device ID is 0x423
- For STM32F401xx devices, the device ID is 0x433
- For STM32F411xx devices, the device ID is 0x431
- For STM32F410xx devices, the device ID is 0x458
- For STM32F412xx devices, the device ID is 0x441
- For STM32F413xx and STM32423xx devices, the device ID is 0x463
- For STM32F446xx devices, the device ID is 0x421
- For STM32F469xx and STM32F479xx devices, the device ID is 0x434

Reference Manual to LL API cross reference:

- DBGMCU_IDCODE DEV_ID LL_DBGMCU_GetDeviceID

LL_DBGMCU_GetRevisionID

Function name

__STATIC_INLINE uint32_t LL_DBGMCU_GetRevisionID (void)

Function description

Return the device revision identifier.

Return values

- **Values:** between Min_Data=0x00 and Max_Data=0xFFFF

Notes

- This field indicates the revision of the device. For example, it is read as RevA -> 0x1000, Cat 2 revZ -> 0x1001, rev1 -> 0x1003, rev2 ->0x1007, revY -> 0x100F for STM32F405/407xx and STM32F415/417xx devices For example, it is read as RevA -> 0x1000, Cat 2 revY -> 0x1003, rev1 -> 0x1007, rev3 ->0x2001 for STM32F42xxx and STM32F43xxx devices For example, it is read as RevZ -> 0x1000, Cat 2 revA -> 0x1001 for STM32F401xB/C devices For example, it is read as RevA -> 0x1000, Cat 2 revZ -> 0x1001 for STM32F401xD/E devices For example, it is read as RevA -> 0x1000 for STM32F411xx,STM32F413/423xx,STM32F469/423xx, STM32F446xx and STM32F410xx devices For example, it is read as RevZ -> 0x1001, Cat 2 revB -> 0x2000, revC -> 0x3000 for STM32F412xx devices

Reference Manual to LL API cross reference:

- DBGMCU_IDCODE REV_ID LL_DBGMCU_GetRevisionID

LL_DBGMCU_EnableDBGSleepMode

Function name

__STATIC_INLINE void LL_DBGMCU_EnableDBGSleepMode (void)

Function description

Enable the Debug Module during SLEEP mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_CR DBG_SLEEP LL_DBGMCU_EnableDBGSleepMode

LL_DBGMCU_DisableDBGSleepMode

Function name

__STATIC_INLINE void LL_DBGMCU_DisableDBGSleepMode (void)

Function description

Disable the Debug Module during SLEEP mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_CR DBG_SLEEP LL_DBGMCU_DisableDBGSleepMode

LL_DBGMCU_EnableDBGStopMode

Function name

__STATIC_INLINE void LL_DBGMCU_EnableDBGStopMode (void)

Function description

Enable the Debug Module during STOP mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_CR DBG_STOP LL_DBGMCU_EnableDBGStopMode

LL_DBGMCU_DisableDBGStopMode

Function name

__STATIC_INLINE void LL_DBGMCU_DisableDBGStopMode (void)

Function description

Disable the Debug Module during STOP mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_CR DBG_STOP LL_DBGMCU_DisableDBGStopMode

LL_DBGMCU_EnableDBGStandbyMode

Function name

__STATIC_INLINE void LL_DBGMCU_EnableDBGStandbyMode (void)

Function description

Enable the Debug Module during STANDBY mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_CR DBG_STANDBY LL_DBGMCU_EnableDBGStandbyMode

LL_DBGMCU_DisableDBGStandbyMode

Function name

__STATIC_INLINE void LL_DBGMCU_DisableDBGStandbyMode (void)

Function description

Disable the Debug Module during STANDBY mode.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_CR DBG_STANDBY LL_DBGMCU_DisableDBGStandbyMode

LL_DBGMCU_SetTracePinAssignment

Function name

__STATIC_INLINE void LL_DBGMCU_SetTracePinAssignment (uint32_t PinAssignment)

Function description

Set Trace pin assignment control.

Parameters

- **PinAssignment:** This parameter can be one of the following values:
  - LL_DBGMCU_TRACE_NONE
  - LL_DBGMCU_TRACE_ASYNCH
  - LL_DBGMCU_TRACE_SYNCH_SIZE1
  - LL_DBGMCU_TRACE_SYNCH_SIZE2
  - LL_DBGMCU_TRACE_SYNCH_SIZE4

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_CR TRACE_IOEN LL_DBGMCU_SetTracePinAssignment
- DBGMCU_CR TRACE_MODE LL_DBGMCU_SetTracePinAssignment

LL_DBGMCU_GetTracePinAssignment

Function name

__STATIC_INLINE uint32_t LL_DBGMCU_GetTracePinAssignment (void)

Function description

Get Trace pin assignment control.

Return values

- **Returned:** value can be one of the following values:
  - LL_DBGMCU_TRACE_NONE
  - LL_DBGMCU_TRACE_ASYNCH
  - LL_DBGMCU_TRACE_SYNCH_SIZE1
  - LL_DBGMCU_TRACE_SYNCH_SIZE2
  - LL_DBGMCU_TRACE_SYNCH_SIZE4

Reference Manual to LL API cross reference:

- DBGMCU_CR TRACE_IOEN LL_DBGMCU_GetTracePinAssignment
- DBGMCU_CR TRACE_MODE LL_DBGMCU_GetTracePinAssignment

LL_DBGMCU_APB1_GRP1_FreezePeriph

Function name

__STATIC_INLINE void LL_DBGMCU_APB1_GRP1_FreezePeriph (uint32_t Periphs)

Function description

Freeze APB1 peripherals (group1 peripherals)

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_DBGMCU_APB1_GRP1_TIM2_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM3_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM4_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM5_STOP
  - LL_DBGMCU_APB1_GRP1_TIM6_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM7_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM12_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM13_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM14_STOP (*)
  - LL_DBGMCU_APB1_GRP1_LPTIM_STOP (*)
  - LL_DBGMCU_APB1_GRP1_RTC_STOP
  - LL_DBGMCU_APB1_GRP1_WWDG_STOP
  - LL_DBGMCU_APB1_GRP1_IWDG_STOP
  - LL_DBGMCU_APB1_GRP1_I2C1_STOP
  - LL_DBGMCU_APB1_GRP1_I2C2_STOP
  - LL_DBGMCU_APB1_GRP1_I2C3_STOP (*)
  - LL_DBGMCU_APB1_GRP1_I2C4_STOP (*)
  - LL_DBGMCU_APB1_GRP1_CAN1_STOP (*)
  - LL_DBGMCU_APB1_GRP1_CAN2_STOP (*)
  - LL_DBGMCU_APB1_GRP1_CAN3_STOP (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_APB1_FZ DBG_TIM2_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_TIM3_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_TIM4_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_TIM5_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_TIM6_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_TIM7_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_TIM12_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_TIM13_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_TIM14_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_LPTIM_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_RTC_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_WWDG_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_IWDG_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_I2C1_SMBUS_TIMEOUT LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_I2C2_SMBUS_TIMEOUT LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_I2C3_SMBUS_TIMEOUT LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_I2C4_SMBUS_TIMEOUT LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_CAN1_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_CAN2_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph
- DBGMCU_APB1_FZ DBG_CAN3_STOP LL_DBGMCU_APB1_GRP1_FreezePeriph

LL_DBGMCU_APB1_GRP1_UnFreezePeriph

Function name

__STATIC_INLINE void LL_DBGMCU_APB1_GRP1_UnFreezePeriph (uint32_t Periphs)

Function description

Unfreeze APB1 peripherals (group1 peripherals)

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_DBGMCU_APB1_GRP1_TIM2_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM3_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM4_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM5_STOP
  - LL_DBGMCU_APB1_GRP1_TIM6_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM7_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM12_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM13_STOP (*)
  - LL_DBGMCU_APB1_GRP1_TIM14_STOP (*)
  - LL_DBGMCU_APB1_GRP1_LPTIM_STOP (*)
  - LL_DBGMCU_APB1_GRP1_RTC_STOP
  - LL_DBGMCU_APB1_GRP1_WWDG_STOP
  - LL_DBGMCU_APB1_GRP1_IWDG_STOP
  - LL_DBGMCU_APB1_GRP1_I2C1_STOP
  - LL_DBGMCU_APB1_GRP1_I2C2_STOP
  - LL_DBGMCU_APB1_GRP1_I2C3_STOP (*)
  - LL_DBGMCU_APB1_GRP1_I2C4_STOP (*)
  - LL_DBGMCU_APB1_GRP1_CAN1_STOP (*)
  - LL_DBGMCU_APB1_GRP1_CAN2_STOP (*)
  - LL_DBGMCU_APB1_GRP1_CAN3_STOP (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_APB1_FZ DBG_TIM2_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_TIM3_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_TIM4_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_TIM5_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_TIM6_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_TIM7_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_TIM12_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_TIM13_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_TIM14_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_LPTIM_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_RTC_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_WWDG_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_IWDG_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_I2C1_SMBUS_TIMEOUT LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_I2C2_SMBUS_TIMEOUT LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_I2C3_SMBUS_TIMEOUT LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_I2C4_SMBUS_TIMEOUT LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_CAN1_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_CAN2_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph
- DBGMCU_APB1_FZ DBG_CAN3_STOP LL_DBGMCU_APB1_GRP1_UnFreezePeriph

LL_DBGMCU_APB2_GRP1_FreezePeriph

Function name

__STATIC_INLINE void LL_DBGMCU_APB2_GRP1_FreezePeriph (uint32_t Periphs)

Function description

Freeze APB2 peripherals.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_DBGMCU_APB2_GRP1_TIM1_STOP
  - LL_DBGMCU_APB2_GRP1_TIM8_STOP (*)
  - LL_DBGMCU_APB2_GRP1_TIM9_STOP (*)
  - LL_DBGMCU_APB2_GRP1_TIM10_STOP (*)
  - LL_DBGMCU_APB2_GRP1_TIM11_STOP (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_APB2_FZ DBG_TIM1_STOP LL_DBGMCU_APB2_GRP1_FreezePeriph
- DBGMCU_APB2_FZ DBG_TIM8_STOP LL_DBGMCU_APB2_GRP1_FreezePeriph
- DBGMCU_APB2_FZ DBG_TIM9_STOP LL_DBGMCU_APB2_GRP1_FreezePeriph
- DBGMCU_APB2_FZ DBG_TIM10_STOP LL_DBGMCU_APB2_GRP1_FreezePeriph
- DBGMCU_APB2_FZ DBG_TIM11_STOP LL_DBGMCU_APB2_GRP1_FreezePeriph

LL_DBGMCU_APB2_GRP1_UnFreezePeriph

Function name

__STATIC_INLINE void LL_DBGMCU_APB2_GRP1_UnFreezePeriph (uint32_t Periphs)

Function description

Unfreeze APB2 peripherals.

Parameters

- **Periphs:** This parameter can be a combination of the following values:
  - LL_DBGMCU_APB2_GRP1_TIM1_STOP
  - LL_DBGMCU_APB2_GRP1_TIM8_STOP (*)
  - LL_DBGMCU_APB2_GRP1_TIM9_STOP (*)
  - LL_DBGMCU_APB2_GRP1_TIM10_STOP (*)
  - LL_DBGMCU_APB2_GRP1_TIM11_STOP (*)

  (*) value not defined in all devices.

Return values

- **None:**

Reference Manual to LL API cross reference:

- DBGMCU_APB2_FZ DBG_TIM1_STOP LL_DBGMCU_APB2_GRP1_UnFreezePeriph
- DBGMCU_APB2_FZ DBG_TIM8_STOP LL_DBGMCU_APB2_GRP1_UnFreezePeriph
- DBGMCU_APB2_FZ DBG_TIM9_STOP LL_DBGMCU_APB2_GRP1_UnFreezePeriph
- DBGMCU_APB2_FZ DBG_TIM10_STOP LL_DBGMCU_APB2_GRP1_UnFreezePeriph
- DBGMCU_APB2_FZ DBG_TIM11_STOP LL_DBGMCU_APB2_GRP1_UnFreezePeriph

LL_FLASH_SetLatency

Function name

__STATIC_INLINE void LL_FLASH_SetLatency (uint32_t Latency)

Function description

Set FLASH Latency.

Parameters

- **Latency:** This parameter can be one of the following values:
  - LL_FLASH_LATENCY_0
  - LL_FLASH_LATENCY_1
  - LL_FLASH_LATENCY_2
  - LL_FLASH_LATENCY_3
  - LL_FLASH_LATENCY_4
  - LL_FLASH_LATENCY_5
  - LL_FLASH_LATENCY_6
  - LL_FLASH_LATENCY_7
  - LL_FLASH_LATENCY_8
  - LL_FLASH_LATENCY_9
  - LL_FLASH_LATENCY_10
  - LL_FLASH_LATENCY_11
  - LL_FLASH_LATENCY_12
  - LL_FLASH_LATENCY_13
  - LL_FLASH_LATENCY_14
  - LL_FLASH_LATENCY_15

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR LATENCY LL_FLASH_SetLatency

LL_FLASH_GetLatency

Function name

__STATIC_INLINE uint32_t LL_FLASH_GetLatency (void)

Function description

Get FLASH Latency.

Return values

- **Returned:** value can be one of the following values:
  - LL_FLASH_LATENCY_0
  - LL_FLASH_LATENCY_1
  - LL_FLASH_LATENCY_2
  - LL_FLASH_LATENCY_3
  - LL_FLASH_LATENCY_4
  - LL_FLASH_LATENCY_5
  - LL_FLASH_LATENCY_6
  - LL_FLASH_LATENCY_7
  - LL_FLASH_LATENCY_8
  - LL_FLASH_LATENCY_9
  - LL_FLASH_LATENCY_10
  - LL_FLASH_LATENCY_11
  - LL_FLASH_LATENCY_12
  - LL_FLASH_LATENCY_13
  - LL_FLASH_LATENCY_14
  - LL_FLASH_LATENCY_15

Reference Manual to LL API cross reference:

- FLASH_ACR LATENCY LL_FLASH_GetLatency

LL_FLASH_EnablePrefetch

Function name

__STATIC_INLINE void LL_FLASH_EnablePrefetch (void)

Function description

Enable Prefetch.

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR PRFTEN LL_FLASH_EnablePrefetch

LL_FLASH_DisablePrefetch

Function name

__STATIC_INLINE void LL_FLASH_DisablePrefetch (void)

Function description

Disable Prefetch.

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR PRFTEN LL_FLASH_DisablePrefetch

LL_FLASH_IsPrefetchEnabled

Function name

__STATIC_INLINE uint32_t LL_FLASH_IsPrefetchEnabled (void)

Function description

Check if Prefetch buffer is enabled.

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- FLASH_ACR PRFTEN LL_FLASH_IsPrefetchEnabled

LL_FLASH_EnableInstCache

Function name

__STATIC_INLINE void LL_FLASH_EnableInstCache (void)

Function description

Enable Instruction cache.

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR ICEN LL_FLASH_EnableInstCache

LL_FLASH_DisableInstCache

Function name

__STATIC_INLINE void LL_FLASH_DisableInstCache (void)

Function description

Disable Instruction cache.

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR ICEN LL_FLASH_DisableInstCache

LL_FLASH_EnableDataCache

Function name

__STATIC_INLINE void LL_FLASH_EnableDataCache (void)

Function description

Enable Data cache.

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR DCEN LL_FLASH_EnableDataCache

LL_FLASH_DisableDataCache

Function name

__STATIC_INLINE void LL_FLASH_DisableDataCache (void)

Function description

Disable Data cache.

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR DCEN LL_FLASH_DisableDataCache

LL_FLASH_EnableInstCacheReset

Function name

__STATIC_INLINE void LL_FLASH_EnableInstCacheReset (void)

Function description

Enable Instruction cache reset.

Return values

- **None:**

Notes

- bit can be written only when the instruction cache is disabled

Reference Manual to LL API cross reference:

- FLASH_ACR ICRST LL_FLASH_EnableInstCacheReset

LL_FLASH_DisableInstCacheReset

Function name

__STATIC_INLINE void LL_FLASH_DisableInstCacheReset (void)

Function description

Disable Instruction cache reset.

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR ICRST LL_FLASH_DisableInstCacheReset

LL_FLASH_EnableDataCacheReset

Function name

__STATIC_INLINE void LL_FLASH_EnableDataCacheReset (void)

Function description

Enable Data cache reset.

Return values

- **None:**

Notes

- bit can be written only when the data cache is disabled

Reference Manual to LL API cross reference:

- FLASH_ACR DCRST LL_FLASH_EnableDataCacheReset

LL_FLASH_DisableDataCacheReset

Function name

__STATIC_INLINE void LL_FLASH_DisableDataCacheReset (void)

Function description

Disable Data cache reset.

Return values

- **None:**

Reference Manual to LL API cross reference:

- FLASH_ACR DCRST LL_FLASH_DisableDataCacheReset

91.2 SYSTEM Firmware driver defines

The following section lists the various define and macros of the module.

91.2.1 SYSTEM

SYSTEM

DBGMCU APB1 GRP1 STOP IP

LL_DBGMCU_APB1_GRP1_TIM2_STOP

TIM2 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_TIM3_STOP

TIM3 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_TIM4_STOP

TIM4 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_TIM5_STOP

TIM5 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_TIM6_STOP

TIM6 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_TIM7_STOP

TIM7 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_TIM12_STOP

TIM12 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_TIM13_STOP

TIM13 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_TIM14_STOP

TIM14 counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_RTC_STOP

RTC counter stopped when core is halted

LL_DBGMCU_APB1_GRP1_WWDG_STOP

Debug Window Watchdog stopped when Core is halted

LL_DBGMCU_APB1_GRP1_IWDG_STOP

Debug Independent Watchdog stopped when Core is halted

LL_DBGMCU_APB1_GRP1_I2C1_STOP

I2C1 SMBUS timeout mode stopped when Core is halted

LL_DBGMCU_APB1_GRP1_I2C2_STOP

I2C2 SMBUS timeout mode stopped when Core is halted

LL_DBGMCU_APB1_GRP1_I2C3_STOP

I2C3 SMBUS timeout mode stopped when Core is halted

LL_DBGMCU_APB1_GRP1_I2C4_STOP

I2C4 SMBUS timeout mode stopped when Core is halted

LL_DBGMCU_APB1_GRP1_CAN1_STOP

CAN1 debug stopped when Core is halted

LL_DBGMCU_APB1_GRP1_CAN2_STOP

CAN2 debug stopped when Core is halted

DBGMCU APB2 GRP1 STOP IP

LL_DBGMCU_APB2_GRP1_TIM1_STOP

TIM1 counter stopped when core is halted

LL_DBGMCU_APB2_GRP1_TIM8_STOP

TIM8 counter stopped when core is halted

LL_DBGMCU_APB2_GRP1_TIM9_STOP

TIM9 counter stopped when core is halted

LL_DBGMCU_APB2_GRP1_TIM10_STOP

TIM10 counter stopped when core is halted

LL_DBGMCU_APB2_GRP1_TIM11_STOP

TIM11 counter stopped when core is halted

SYSCFG BANK MODE

LL_SYSCFG_BANKMODE_BANK1

Flash Bank 1 base address mapped at 0x0800 0000 (AXI) and 0x0020 0000 (TCM) and Flash Bank 2 base address mapped at 0x0810 0000 (AXI) and 0x0030 0000 (TCM)

LL_SYSCFG_BANKMODE_BANK2

Flash Bank 2 base address mapped at 0x0800 0000 (AXI) and 0x0020 0000(TCM) and Flash Bank 1 base address mapped at 0x0810 0000 (AXI) and 0x0030 0000(TCM)

SYSCFG EXTI LINE

LL_SYSCFG_EXTI_LINE0

EXTI_POSITION_0 | EXTICR[0]

LL_SYSCFG_EXTI_LINE1

EXTI_POSITION_4 | EXTICR[0]

LL_SYSCFG_EXTI_LINE2

EXTI_POSITION_8 | EXTICR[0]

LL_SYSCFG_EXTI_LINE3

EXTI_POSITION_12 | EXTICR[0]

LL_SYSCFG_EXTI_LINE4

EXTI_POSITION_0 | EXTICR[1]

LL_SYSCFG_EXTI_LINE5

EXTI_POSITION_4 | EXTICR[1]

LL_SYSCFG_EXTI_LINE6

EXTI_POSITION_8 | EXTICR[1]

LL_SYSCFG_EXTI_LINE7

EXTI_POSITION_12 | EXTICR[1]

LL_SYSCFG_EXTI_LINE8

EXTI_POSITION_0 | EXTICR[2]

LL_SYSCFG_EXTI_LINE9

EXTI_POSITION_4 | EXTICR[2]

LL_SYSCFG_EXTI_LINE10

EXTI_POSITION_8 | EXTICR[2]

LL_SYSCFG_EXTI_LINE11

EXTI_POSITION_12 | EXTICR[2]

LL_SYSCFG_EXTI_LINE12

EXTI_POSITION_0 | EXTICR[3]

LL_SYSCFG_EXTI_LINE13

EXTI_POSITION_4 | EXTICR[3]

LL_SYSCFG_EXTI_LINE14

EXTI_POSITION_8 | EXTICR[3]

LL_SYSCFG_EXTI_LINE15

EXTI_POSITION_12 | EXTICR[3]

SYSCFG EXTI PORT

LL_SYSCFG_EXTI_PORTA

EXTI PORT A

LL_SYSCFG_EXTI_PORTB

EXTI PORT B

LL_SYSCFG_EXTI_PORTC

EXTI PORT C

LL_SYSCFG_EXTI_PORTD

EXTI PORT D

LL_SYSCFG_EXTI_PORTE

EXTI PORT E

LL_SYSCFG_EXTI_PORTF

EXTI PORT F

LL_SYSCFG_EXTI_PORTG

EXTI PORT G

LL_SYSCFG_EXTI_PORTH

EXTI PORT H

LL_SYSCFG_EXTI_PORTI

EXTI PORT I

LL_SYSCFG_EXTI_PORTJ

EXTI PORT J

LL_SYSCFG_EXTI_PORTK

EXTI PORT k

FLASH LATENCY

LL_FLASH_LATENCY_0

FLASH Zero wait state

LL_FLASH_LATENCY_1

FLASH One wait state

LL_FLASH_LATENCY_2

FLASH Two wait states

LL_FLASH_LATENCY_3

FLASH Three wait states

LL_FLASH_LATENCY_4

FLASH Four wait states

LL_FLASH_LATENCY_5

FLASH five wait state

LL_FLASH_LATENCY_6

FLASH six wait state

LL_FLASH_LATENCY_7

FLASH seven wait states

LL_FLASH_LATENCY_8

FLASH eight wait states

LL_FLASH_LATENCY_9

FLASH nine wait states

LL_FLASH_LATENCY_10

FLASH ten wait states

LL_FLASH_LATENCY_11

FLASH eleven wait states

LL_FLASH_LATENCY_12

FLASH twelve wait states

LL_FLASH_LATENCY_13

FLASH thirteen wait states

LL_FLASH_LATENCY_14

FLASH fourteen wait states

LL_FLASH_LATENCY_15

FLASH fifteen wait states

SYSCFG PMC

LL_SYSCFG_PMC_ETHMII

ETH Media MII interface

LL_SYSCFG_PMC_ETHRMII

ETH Media RMII interface

SYSCFG REMAP

LL_SYSCFG_REMAP_FLASH

Main Flash memory mapped at 0x00000000

LL_SYSCFG_REMAP_SYSTEMFLASH

System Flash memory mapped at 0x00000000

LL_SYSCFG_REMAP_FMC

FMC(NOR/PSRAM 1 and 2) mapped at 0x00000000

LL_SYSCFG_REMAP_SDRAM

FMC/SDRAM mapped at 0x00000000

LL_SYSCFG_REMAP_SRAM

SRAM1 mapped at 0x00000000

DBGMCU TRACE Pin Assignment

LL_DBGMCU_TRACE_NONE

TRACE pins not assigned (default state)

LL_DBGMCU_TRACE_ASYNCH

TRACE pin assignment for Asynchronous Mode

LL_DBGMCU_TRACE_SYNCH_SIZE1

TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1

LL_DBGMCU_TRACE_SYNCH_SIZE2

TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2

LL_DBGMCU_TRACE_SYNCH_SIZE4

TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

92 LL TIM Generic Driver

92.1 TIM Firmware driver registers structures

92.1.1 LL_TIM_InitTypeDef

LL_TIM_InitTypeDef is defined in the stm32f4xx_ll_tim.h

Data Fields

- ***uint16_t Prescaler***
- ***uint32_t CounterMode***
- ***uint32_t Autoreload***
- ***uint32_t ClockDivision***
- ***uint32_t RepetitionCounter***

Field Documentation

- ***uint16_t LL_TIM_InitTypeDef::Prescaler***

  Specifies the prescaler value used to divide the TIM clock. This parameter can be a number between Min_Data=0x0000 and Max_Data=0xFFFF.This feature can be modified afterwards using unitary function **LL_TIM_SetPrescaler()**.
- ***uint32_t LL_TIM_InitTypeDef::CounterMode***

  Specifies the counter mode. This parameter can be a value of ***TIM_LL_EC_COUNTERMODE***.This feature can be modified afterwards using unitary function **LL_TIM_SetCounterMode()**.
- ***uint32_t LL_TIM_InitTypeDef::Autoreload***

  Specifies the auto reload value to be loaded into the active Auto-Reload Register at the next update event. This parameter must be a number between Min_Data=0x0000 and Max_Data=0xFFFF. Some timer instances may support 32 bits counters. In that case this parameter must be a number between 0x0000 and 0xFFFFFFFF.This feature can be modified afterwards using unitary function **LL_TIM_SetAutoReload()**.
- ***uint32_t LL_TIM_InitTypeDef::ClockDivision***

  Specifies the clock division. This parameter can be a value of ***TIM_LL_EC_CLOCKDIVISION***.This feature can be modified afterwards using unitary function **LL_TIM_SetClockDivision()**.
- ***uint32_t LL_TIM_InitTypeDef::RepetitionCounter***

  Specifies the repetition counter value. Each time the RCR downcounter reaches zero, an update event is generated and counting restarts from the RCR value (N). This means in PWM mode that (N+1) corresponds to:
  - the number of PWM periods in edge-aligned mode
  - the number of half PWM period in center-aligned mode GP timers: this parameter must be a number between Min_Data = 0x00 and Max_Data = 0xFF. Advanced timers: this parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF.

  This feature can be modified afterwards using unitary function **LL_TIM_SetRepetitionCounter()**.

92.1.2 LL_TIM_OC_InitTypeDef

LL_TIM_OC_InitTypeDef is defined in the stm32f4xx_ll_tim.h

Data Fields

- ***uint32_t OCMode***
- ***uint32_t OCState***
- ***uint32_t OCNState***
- ***uint32_t CompareValue***
- ***uint32_t OCPolarity***
- ***uint32_t OCNPolarity***
- ***uint32_t OCIdleState***
- ***uint32_t OCNIdleState***

Field Documentation

- ***uint32_t LL_TIM_OC_InitTypeDef::OCMode***

  Specifies the output mode. This parameter can be a value of ***TIM_LL_EC_OCMODE***.This feature can be modified afterwards using unitary function **LL_TIM_OC_SetMode()**.
- ***uint32_t LL_TIM_OC_InitTypeDef::OCState***

  Specifies the TIM Output Compare state. This parameter can be a value of ***TIM_LL_EC_OCSTATE***.This feature can be modified afterwards using unitary functions **LL_TIM_CC_EnableChannel()** or **LL_TIM_CC_DisableChannel()**.
- ***uint32_t LL_TIM_OC_InitTypeDef::OCNState***

  Specifies the TIM complementary Output Compare state. This parameter can be a value of ***TIM_LL_EC_OCSTATE***.This feature can be modified afterwards using unitary functions **LL_TIM_CC_EnableChannel()** or **LL_TIM_CC_DisableChannel()**.
- ***uint32_t LL_TIM_OC_InitTypeDef::CompareValue***

  Specifies the Compare value to be loaded into the Capture Compare Register. This parameter can be a number between Min_Data=0x0000 and Max_Data=0xFFFF.This feature can be modified afterwards using unitary function LL_TIM_OC_SetCompareCHx (x=1..6).
- ***uint32_t LL_TIM_OC_InitTypeDef::OCPolarity***

  Specifies the output polarity. This parameter can be a value of ***TIM_LL_EC_OCPOLARITY***.This feature can be modified afterwards using unitary function **LL_TIM_OC_SetPolarity()**.
- ***uint32_t LL_TIM_OC_InitTypeDef::OCNPolarity***

  Specifies the complementary output polarity. This parameter can be a value of ***TIM_LL_EC_OCPOLARITY***.This feature can be modified afterwards using unitary function **LL_TIM_OC_SetPolarity()**.
- ***uint32_t LL_TIM_OC_InitTypeDef::OCIdleState***

  Specifies the TIM Output Compare pin state during Idle state. This parameter can be a value of ***TIM_LL_EC_OCIDLESTATE***.This feature can be modified afterwards using unitary function **LL_TIM_OC_SetIdleState()**.
- ***uint32_t LL_TIM_OC_InitTypeDef::OCNIdleState***

  Specifies the TIM Output Compare pin state during Idle state. This parameter can be a value of ***TIM_LL_EC_OCIDLESTATE***.This feature can be modified afterwards using unitary function **LL_TIM_OC_SetIdleState()**.

92.1.3 LL_TIM_IC_InitTypeDef

LL_TIM_IC_InitTypeDef is defined in the stm32f4xx_ll_tim.h

Data Fields

- ***uint32_t ICPolarity***
- ***uint32_t ICActiveInput***
- ***uint32_t ICPrescaler***
- ***uint32_t ICFilter***

Field Documentation

- ***uint32_t LL_TIM_IC_InitTypeDef::ICPolarity***

  Specifies the active edge of the input signal. This parameter can be a value of ***TIM_LL_EC_IC_POLARITY***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPolarity()**.
- ***uint32_t LL_TIM_IC_InitTypeDef::ICActiveInput***

  Specifies the input. This parameter can be a value of ***TIM_LL_EC_ACTIVEINPUT***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetActiveInput()**.
- ***uint32_t LL_TIM_IC_InitTypeDef::ICPrescaler***

  Specifies the Input Capture Prescaler. This parameter can be a value of ***TIM_LL_EC_ICPSC***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPrescaler()**.
- ***uint32_t LL_TIM_IC_InitTypeDef::ICFilter***

  Specifies the input capture filter. This parameter can be a value of ***TIM_LL_EC_IC_FILTER***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetFilter()**.

92.1.4 LL_TIM_ENCODER_InitTypeDef

LL_TIM_ENCODER_InitTypeDef is defined in the stm32f4xx_ll_tim.h

Data Fields

- ***uint32_t EncoderMode***
- ***uint32_t IC1Polarity***
- ***uint32_t IC1ActiveInput***
- ***uint32_t IC1Prescaler***
- ***uint32_t IC1Filter***
- ***uint32_t IC2Polarity***
- ***uint32_t IC2ActiveInput***
- ***uint32_t IC2Prescaler***
- ***uint32_t IC2Filter***

Field Documentation

- ***uint32_t LL_TIM_ENCODER_InitTypeDef::EncoderMode***

  Specifies the encoder resolution (x2 or x4). This parameter can be a value of ***TIM_LL_EC_ENCODERMODE***.This feature can be modified afterwards using unitary function **LL_TIM_SetEncoderMode()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC1Polarity***

  Specifies the active edge of TI1 input. This parameter can be a value of ***TIM_LL_EC_IC_POLARITY***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPolarity()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC1ActiveInput***

  Specifies the TI1 input source This parameter can be a value of ***TIM_LL_EC_ACTIVEINPUT***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetActiveInput()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC1Prescaler***

  Specifies the TI1 input prescaler value. This parameter can be a value of ***TIM_LL_EC_ICPSC***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPrescaler()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC1Filter***

  Specifies the TI1 input filter. This parameter can be a value of ***TIM_LL_EC_IC_FILTER***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetFilter()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC2Polarity***

  Specifies the active edge of TI2 input. This parameter can be a value of ***TIM_LL_EC_IC_POLARITY***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPolarity()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC2ActiveInput***

  Specifies the TI2 input source This parameter can be a value of ***TIM_LL_EC_ACTIVEINPUT***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetActiveInput()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC2Prescaler***

  Specifies the TI2 input prescaler value. This parameter can be a value of ***TIM_LL_EC_ICPSC***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPrescaler()**.
- ***uint32_t LL_TIM_ENCODER_InitTypeDef::IC2Filter***

  Specifies the TI2 input filter. This parameter can be a value of ***TIM_LL_EC_IC_FILTER***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetFilter()**.

92.1.5 LL_TIM_HALLSENSOR_InitTypeDef

LL_TIM_HALLSENSOR_InitTypeDef is defined in the stm32f4xx_ll_tim.h

Data Fields

- ***uint32_t IC1Polarity***
- ***uint32_t IC1Prescaler***
- ***uint32_t IC1Filter***
- ***uint32_t CommutationDelay***

Field Documentation

- ***uint32_t LL_TIM_HALLSENSOR_InitTypeDef::IC1Polarity***

  Specifies the active edge of TI1 input. This parameter can be a value of ***TIM_LL_EC_IC_POLARITY***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPolarity()**.
- ***uint32_t LL_TIM_HALLSENSOR_InitTypeDef::IC1Prescaler***

  Specifies the TI1 input prescaler value. Prescaler must be set to get a maximum counter period longer than the time interval between 2 consecutive changes on the Hall inputs. This parameter can be a value of ***TIM_LL_EC_ICPSC***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetPrescaler()**.
- ***uint32_t LL_TIM_HALLSENSOR_InitTypeDef::IC1Filter***

  Specifies the TI1 input filter. This parameter can be a value of ***TIM_LL_EC_IC_FILTER***.This feature can be modified afterwards using unitary function **LL_TIM_IC_SetFilter()**.
- ***uint32_t LL_TIM_HALLSENSOR_InitTypeDef::CommutationDelay***

  Specifies the compare value to be loaded into the Capture Compare Register. A positive pulse (TRGO event) is generated with a programmable delay every time a change occurs on the Hall inputs. This parameter can be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF.This feature can be modified afterwards using unitary function **LL_TIM_OC_SetCompareCH2()**.

92.1.6 LL_TIM_BDTR_InitTypeDef

LL_TIM_BDTR_InitTypeDef is defined in the stm32f4xx_ll_tim.h

Data Fields

- ***uint32_t OSSRState***
- ***uint32_t OSSIState***
- ***uint32_t LockLevel***
- ***uint8_t DeadTime***
- ***uint16_t BreakState***
- ***uint32_t BreakPolarity***
- ***uint32_t AutomaticOutput***

Field Documentation

- ***uint32_t LL_TIM_BDTR_InitTypeDef::OSSRState***

  Specifies the Off-State selection used in Run mode. This parameter can be a value of ***TIM_LL_EC_OSSR***This feature can be modified afterwards using unitary function **LL_TIM_SetOffStates()**

  Note:

  – This bit-field cannot be modified as long as LOCK level 2 has been programmed.
- ***uint32_t LL_TIM_BDTR_InitTypeDef::OSSIState***

  Specifies the Off-State used in Idle state. This parameter can be a value of ***TIM_LL_EC_OSSI***This feature can be modified afterwards using unitary function **LL_TIM_SetOffStates()**

  Note:

  – This bit-field cannot be modified as long as LOCK level 2 has been programmed.
- ***uint32_t LL_TIM_BDTR_InitTypeDef::LockLevel***

  Specifies the LOCK level parameters. This parameter can be a value of ***TIM_LL_EC_LOCKLEVEL***

  Note:

  – The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.
- ***uint8_t LL_TIM_BDTR_InitTypeDef::DeadTime***

  Specifies the delay time between the switching-off and the switching-on of the outputs. This parameter can be a number between Min_Data = 0x00 and Max_Data = 0xFF.This feature can be modified afterwards using unitary function **LL_TIM_OC_SetDeadTime()**

  Note:

  – This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed.

- ***uint16_t LL_TIM_BDTR_InitTypeDef::BreakState***

  Specifies whether the TIM Break input is enabled or not. This parameter can be a value of ***TIM_LL_EC_BREAK_ENABLE***This feature can be modified afterwards using unitary functions **LL_TIM_EnableBRK()** or **LL_TIM_DisableBRK()**

  Note:

  – This bit-field can not be modified as long as LOCK level 1 has been programmed.

- ***uint32_t LL_TIM_BDTR_InitTypeDef::BreakPolarity***

  Specifies the TIM Break Input pin polarity. This parameter can be a value of ***TIM_LL_EC_BREAK_POLARITY***This feature can be modified afterwards using unitary function **LL_TIM_ConfigBRK()**

  Note:

  – This bit-field can not be modified as long as LOCK level 1 has been programmed.

- ***uint32_t LL_TIM_BDTR_InitTypeDef::AutomaticOutput***

  Specifies whether the TIM Automatic Output feature is enabled or not. This parameter can be a value of ***TIM_LL_EC_AUTOMATICOUTPUT_ENABLE***This feature can be modified afterwards using unitary functions **LL_TIM_EnableAutomaticOutput()** or **LL_TIM_DisableAutomaticOutput()**

  Note:

  – This bit-field can not be modified as long as LOCK level 1 has been programmed.

92.2 TIM Firmware driver API description

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

92.2.1 Detailed description of functions

LL_TIM_EnableCounter

Function name

__STATIC_INLINE void LL_TIM_EnableCounter (TIM_TypeDef * TIMx)

Function description

Enable timer counter.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 CEN LL_TIM_EnableCounter

LL_TIM_DisableCounter

Function name

__STATIC_INLINE void LL_TIM_DisableCounter (TIM_TypeDef * TIMx)

Function description

Disable timer counter.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 CEN LL_TIM_DisableCounter

LL_TIM_IsEnabledCounter

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledCounter (TIM_TypeDef * TIMx)

Function description

Indicates whether the timer counter is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 CEN LL_TIM_IsEnabledCounter

LL_TIM_EnableUpdateEvent

Function name

__STATIC_INLINE void LL_TIM_EnableUpdateEvent (TIM_TypeDef * TIMx)

Function description

Enable update event generation.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 UDIS LL_TIM_EnableUpdateEvent

LL_TIM_DisableUpdateEvent

Function name

__STATIC_INLINE void LL_TIM_DisableUpdateEvent (TIM_TypeDef * TIMx)

Function description

Disable update event generation.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 UDIS LL_TIM_DisableUpdateEvent

LL_TIM_IsEnabledUpdateEvent

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledUpdateEvent (TIM_TypeDef * TIMx)

Function description

Indicates whether update event generation is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **Inverted:** state of bit (0 or 1).

Reference Manual to LL API cross reference:

- CR1 UDIS LL_TIM_IsEnabledUpdateEvent

LL_TIM_SetUpdateSource

Function name

__STATIC_INLINE void LL_TIM_SetUpdateSource (TIM_TypeDef * TIMx, uint32_t UpdateSource)

Function description

Set update event source.

Parameters

- **TIMx:** Timer instance
- **UpdateSource:** This parameter can be one of the following values:
  - LL_TIM_UPDATESOURCE_REGULAR
  - LL_TIM_UPDATESOURCE_COUNTER

Return values

- **None:**

Notes

- Update event source set to LL_TIM_UPDATESOURCE_REGULAR: any of the following events generate an update interrupt or DMA request if enabled: Counter overflow/underflowSetting the UG bitUpdate generation through the slave mode controller
- Update event source set to LL_TIM_UPDATESOURCE_COUNTER: only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Reference Manual to LL API cross reference:

- CR1 URS LL_TIM_SetUpdateSource

LL_TIM_GetUpdateSource

Function name

__STATIC_INLINE uint32_t LL_TIM_GetUpdateSource (TIM_TypeDef * TIMx)

Function description

Get actual event update source.

Parameters

- **TIMx:** Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_UPDATESOURCE_REGULAR
  - LL_TIM_UPDATESOURCE_COUNTER

Reference Manual to LL API cross reference:

- CR1 URS LL_TIM_GetUpdateSource

LL_TIM_SetOnePulseMode

Function name

__STATIC_INLINE void LL_TIM_SetOnePulseMode (TIM_TypeDef * TIMx, uint32_t OnePulseMode)

Function description

Set one pulse mode (one shot v.s.

Parameters

- **TIMx:** Timer instance
- **OnePulseMode:** This parameter can be one of the following values:
  - LL_TIM_ONEPULSEMODE_SINGLE
  - LL_TIM_ONEPULSEMODE_REPETITIVE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 OPM LL_TIM_SetOnePulseMode

LL_TIM_GetOnePulseMode

Function name

__STATIC_INLINE uint32_t LL_TIM_GetOnePulseMode (TIM_TypeDef * TIMx)

Function description

Get actual one pulse mode.

Parameters

- **TIMx:** Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_ONEPULSEMODE_SINGLE
  - LL_TIM_ONEPULSEMODE_REPETITIVE

Reference Manual to LL API cross reference:

- CR1 OPM LL_TIM_GetOnePulseMode

LL_TIM_SetCounterMode

Function name

__STATIC_INLINE void LL_TIM_SetCounterMode (TIM_TypeDef * TIMx, uint32_t CounterMode)

Function description

Set the timer counter counting mode.

Parameters

- **TIMx:** Timer instance
- **CounterMode:** This parameter can be one of the following values:
  - LL_TIM_COUNTERMODE_UP
  - LL_TIM_COUNTERMODE_DOWN
  - LL_TIM_COUNTERMODE_CENTER_UP
  - LL_TIM_COUNTERMODE_CENTER_DOWN
  - LL_TIM_COUNTERMODE_CENTER_UP_DOWN

Return values

- **None:**

Notes

- Macro IS_TIM_COUNTER_MODE_SELECT_INSTANCE(TIMx) can be used to check whether or not the counter mode selection feature is supported by a timer instance.
- Switching from Center Aligned counter mode to Edge counter mode (or reverse) requires a timer reset to avoid unexpected direction due to DIR bit readonly in center aligned mode.

Reference Manual to LL API cross reference:

- CR1 DIR LL_TIM_SetCounterMode
- CR1 CMS LL_TIM_SetCounterMode

LL_TIM_GetCounterMode

Function name

__STATIC_INLINE uint32_t LL_TIM_GetCounterMode (TIM_TypeDef * TIMx)

Function description

Get actual counter mode.

Parameters

- **TIMx:** Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_COUNTERMODE_UP
  - LL_TIM_COUNTERMODE_DOWN
  - LL_TIM_COUNTERMODE_CENTER_UP
  - LL_TIM_COUNTERMODE_CENTER_DOWN
  - LL_TIM_COUNTERMODE_CENTER_UP_DOWN

Notes

- Macro IS_TIM_COUNTER_MODE_SELECT_INSTANCE(TIMx) can be used to check whether or not the counter mode selection feature is supported by a timer instance.

Reference Manual to LL API cross reference:

- CR1 DIR LL_TIM_GetCounterMode
- CR1 CMS LL_TIM_GetCounterMode

LL_TIM_EnableARRPreload

Function name

__STATIC_INLINE void LL_TIM_EnableARRPreload (TIM_TypeDef * TIMx)

Function description

Enable auto-reload (ARR) preload.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 ARPE LL_TIM_EnableARRPreload

LL_TIM_DisableARRPreload

Function name

__STATIC_INLINE void LL_TIM_DisableARRPreload (TIM_TypeDef * TIMx)

Function description

Disable auto-reload (ARR) preload.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 ARPE LL_TIM_DisableARRPreload

LL_TIM_IsEnabledARRPreload

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledARRPreload (TIM_TypeDef * TIMx)

Function description

Indicates whether auto-reload (ARR) preload is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 ARPE LL_TIM_IsEnabledARRPreload

LL_TIM_SetClockDivision

Function name

__STATIC_INLINE void LL_TIM_SetClockDivision (TIM_TypeDef * TIMx, uint32_t ClockDivision)

Function description

Set the division ratio between the timer clock and the sampling clock used by the dead-time generators (when supported) and the digital filters.

Parameters

- **TIMx:** Timer instance
- **ClockDivision:** This parameter can be one of the following values:
  - LL_TIM_CLOCKDIVISION_DIV1
  - LL_TIM_CLOCKDIVISION_DIV2
  - LL_TIM_CLOCKDIVISION_DIV4

Return values

- **None:**

Notes

- Macro IS_TIM_CLOCK_DIVISION_INSTANCE(TIMx) can be used to check whether or not the clock division feature is supported by the timer instance.

Reference Manual to LL API cross reference:

- CR1 CKD LL_TIM_SetClockDivision

LL_TIM_GetClockDivision

Function name

__STATIC_INLINE uint32_t LL_TIM_GetClockDivision (TIM_TypeDef * TIMx)

Function description

Get the actual division ratio between the timer clock and the sampling clock used by the dead-time generators (when supported) and the digital filters.

Parameters

- **TIMx:** Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_CLOCKDIVISION_DIV1
  - LL_TIM_CLOCKDIVISION_DIV2
  - LL_TIM_CLOCKDIVISION_DIV4

Notes

- Macro IS_TIM_CLOCK_DIVISION_INSTANCE(TIMx) can be used to check whether or not the clock division feature is supported by the timer instance.

Reference Manual to LL API cross reference:

- CR1 CKD LL_TIM_GetClockDivision

LL_TIM_SetCounter

Function name

__STATIC_INLINE void LL_TIM_SetCounter (TIM_TypeDef * TIMx, uint32_t Counter)

Function description

Set the counter value.

Parameters

- **TIMx:** Timer instance
- **Counter:** Counter value (between Min_Data=0 and Max_Data=0xFFFF or 0xFFFFFFFF)

Return values

- **None:**

Notes

- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.

Reference Manual to LL API cross reference:

- CNT CNT LL_TIM_SetCounter

LL_TIM_GetCounter

Function name

__STATIC_INLINE uint32_t LL_TIM_GetCounter (TIM_TypeDef * TIMx)

Function description

Get the counter value.

Parameters

- **TIMx:** Timer instance

Return values

- **Counter:** value (between Min_Data=0 and Max_Data=0xFFFF or 0xFFFFFFFF)

Notes

- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.

Reference Manual to LL API cross reference:

- CNT CNT LL_TIM_GetCounter

LL_TIM_GetDirection

Function name

__STATIC_INLINE uint32_t LL_TIM_GetDirection (TIM_TypeDef * TIMx)

Function description

Get the current direction of the counter.

Parameters

- **TIMx:** Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_COUNTERDIRECTION_UP
  - LL_TIM_COUNTERDIRECTION_DOWN

Reference Manual to LL API cross reference:

- CR1 DIR LL_TIM_GetDirection

LL_TIM_SetPrescaler

Function name

__STATIC_INLINE void LL_TIM_SetPrescaler (TIM_TypeDef * TIMx, uint32_t Prescaler)

Function description

Set the prescaler value.

Parameters

- **TIMx:** Timer instance
- **Prescaler:** between Min_Data=0 and Max_Data=65535

Return values

- **None:**

Notes

- The counter clock frequency CK_CNT is equal to fCK_PSC / (PSC[15:0] + 1).
- The prescaler can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.
- Helper macro __LL_TIM_CALC_PSC can be used to calculate the Prescaler parameter

Reference Manual to LL API cross reference:

- PSC PSC LL_TIM_SetPrescaler

LL_TIM_GetPrescaler

Function name

__STATIC_INLINE uint32_t LL_TIM_GetPrescaler (TIM_TypeDef * TIMx)

Function description

Get the prescaler value.

Parameters

- **TIMx:** Timer instance

Return values

- **Prescaler:** value between Min_Data=0 and Max_Data=65535

Reference Manual to LL API cross reference:

- PSC PSC LL_TIM_GetPrescaler

LL_TIM_SetAutoReload

Function name

__STATIC_INLINE void LL_TIM_SetAutoReload (TIM_TypeDef * TIMx, uint32_t AutoReload)

Function description

Set the auto-reload value.

Parameters

- **TIMx:** Timer instance
- **AutoReload:** between Min_Data=0 and Max_Data=65535

Return values

- **None:**

Notes

- The counter is blocked while the auto-reload value is null.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Helper macro __LL_TIM_CALC_ARR can be used to calculate the AutoReload parameter

Reference Manual to LL API cross reference:

- ARR ARR LL_TIM_SetAutoReload

LL_TIM_GetAutoReload

Function name

__STATIC_INLINE uint32_t LL_TIM_GetAutoReload (TIM_TypeDef * TIMx)

Function description

Get the auto-reload value.

Parameters

- **TIMx:** Timer instance

Return values

- **Auto-reload:** value

Notes

- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.

Reference Manual to LL API cross reference:

- ARR ARR LL_TIM_GetAutoReload

LL_TIM_SetRepetitionCounter

Function name

__STATIC_INLINE void LL_TIM_SetRepetitionCounter (TIM_TypeDef * TIMx, uint32_t RepetitionCounter)

Function description

Set the repetition counter value.

Parameters

- **TIMx:** Timer instance
- **RepetitionCounter:** between Min_Data=0 and Max_Data=255 or 65535 for advanced timer.

Return values

- **None:**

Notes

- Macro IS_TIM_REPETITION_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a repetition counter.

Reference Manual to LL API cross reference:

- RCR REP LL_TIM_SetRepetitionCounter

LL_TIM_GetRepetitionCounter

Function name

__STATIC_INLINE uint32_t LL_TIM_GetRepetitionCounter (TIM_TypeDef * TIMx)

Function description

Get the repetition counter value.

Parameters

- **TIMx:** Timer instance

Return values

- **Repetition:** counter value

Notes

- Macro IS_TIM_REPETITION_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a repetition counter.

Reference Manual to LL API cross reference:

- RCR REP LL_TIM_GetRepetitionCounter

LL_TIM_CC_EnablePreload

Function name

__STATIC_INLINE void LL_TIM_CC_EnablePreload (TIM_TypeDef * TIMx)

Function description

Enable the capture/compare control bits (CCxE, CCxNE and OCxM) preload.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs.
- Only on channels that have a complementary output.
- Macro IS_TIM_COMMUTATION_EVENT_INSTANCE(TIMx) can be used to check whether or not a timer instance is able to generate a commutation event.

Reference Manual to LL API cross reference:

- CR2 CCPC LL_TIM_CC_EnablePreload

LL_TIM_CC_DisablePreload

Function name

__STATIC_INLINE void LL_TIM_CC_DisablePreload (TIM_TypeDef * TIMx)

Function description

Disable the capture/compare control bits (CCxE, CCxNE and OCxM) preload.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_COMMUTATION_EVENT_INSTANCE(TIMx) can be used to check whether or not a timer instance is able to generate a commutation event.

Reference Manual to LL API cross reference:

- CR2 CCPC LL_TIM_CC_DisablePreload

LL_TIM_CC_SetUpdate

Function name

__STATIC_INLINE void LL_TIM_CC_SetUpdate (TIM_TypeDef * TIMx, uint32_t CCUpdateSource)

Function description

Set the updated source of the capture/compare control bits (CCxE, CCxNE and OCxM).

Parameters

- **TIMx:** Timer instance
- **CCUpdateSource:** This parameter can be one of the following values:
  - LL_TIM_CCUPDATESOURCE_COMG_ONLY
  - LL_TIM_CCUPDATESOURCE_COMG_AND_TRGI

Return values

- **None:**

Notes

- Macro IS_TIM_COMMUTATION_EVENT_INSTANCE(TIMx) can be used to check whether or not a timer instance is able to generate a commutation event.

Reference Manual to LL API cross reference:

- CR2 CCUS LL_TIM_CC_SetUpdate

LL_TIM_CC_SetDMAReqTrigger

Function name

__STATIC_INLINE void LL_TIM_CC_SetDMAReqTrigger (TIM_TypeDef * TIMx, uint32_t DMAReqTrigger)

Function description

Set the trigger of the capture/compare DMA request.

Parameters

- **TIMx:** Timer instance
- **DMAReqTrigger:** This parameter can be one of the following values:
  - LL_TIM_CCDMAREQUEST_CC
  - LL_TIM_CCDMAREQUEST_UPDATE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 CCDS LL_TIM_CC_SetDMAReqTrigger

LL_TIM_CC_GetDMAReqTrigger

Function name

__STATIC_INLINE uint32_t LL_TIM_CC_GetDMAReqTrigger (TIM_TypeDef * TIMx)

Function description

Get actual trigger of the capture/compare DMA request.

Parameters

- **TIMx:** Timer instance

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_CCDMAREQUEST_CC
  - LL_TIM_CCDMAREQUEST_UPDATE

Reference Manual to LL API cross reference:

- CR2 CCDS LL_TIM_CC_GetDMAReqTrigger

LL_TIM_CC_SetLockLevel

Function name

__STATIC_INLINE void LL_TIM_CC_SetLockLevel (TIM_TypeDef * TIMx, uint32_t LockLevel)

Function description

Set the lock level to freeze the configuration of several capture/compare parameters.

Parameters

- **TIMx:** Timer instance
- **LockLevel:** This parameter can be one of the following values:
  - LL_TIM_LOCKLEVEL_OFF
  - LL_TIM_LOCKLEVEL_1
  - LL_TIM_LOCKLEVEL_2
  - LL_TIM_LOCKLEVEL_3

Return values

- **None:**

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not the lock mechanism is supported by a timer instance.

Reference Manual to LL API cross reference:

- BDTR LOCK LL_TIM_CC_SetLockLevel

LL_TIM_CC_EnableChannel

Function name

__STATIC_INLINE void LL_TIM_CC_EnableChannel (TIM_TypeDef * TIMx, uint32_t Channels)

Function description

Enable capture/compare channels.

Parameters

- **TIMx:** Timer instance
- **Channels:** This parameter can be a combination of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH1N
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH2N
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH3N
  - LL_TIM_CHANNEL_CH4

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCER CC1E LL_TIM_CC_EnableChannel
- CCER CC1NE LL_TIM_CC_EnableChannel
- CCER CC2E LL_TIM_CC_EnableChannel
- CCER CC2NE LL_TIM_CC_EnableChannel
- CCER CC3E LL_TIM_CC_EnableChannel
- CCER CC3NE LL_TIM_CC_EnableChannel
- CCER CC4E LL_TIM_CC_EnableChannel

LL_TIM_CC_DisableChannel

Function name

__STATIC_INLINE void LL_TIM_CC_DisableChannel (TIM_TypeDef * TIMx, uint32_t Channels)

Function description

Disable capture/compare channels.

Parameters

- **TIMx:** Timer instance
- **Channels:** This parameter can be a combination of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH1N
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH2N
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH3N
  - LL_TIM_CHANNEL_CH4

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCER CC1E LL_TIM_CC_DisableChannel
- CCER CC1NE LL_TIM_CC_DisableChannel
- CCER CC2E LL_TIM_CC_DisableChannel
- CCER CC2NE LL_TIM_CC_DisableChannel
- CCER CC3E LL_TIM_CC_DisableChannel
- CCER CC3NE LL_TIM_CC_DisableChannel
- CCER CC4E LL_TIM_CC_DisableChannel

LL_TIM_CC_IsEnabledChannel

Function name

__STATIC_INLINE uint32_t LL_TIM_CC_IsEnabledChannel (TIM_TypeDef * TIMx, uint32_t Channels)

Function description

Indicate whether channel(s) is(are) enabled.

Parameters

- **TIMx:** Timer instance
- **Channels:** This parameter can be a combination of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH1N
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH2N
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH3N
  - LL_TIM_CHANNEL_CH4

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CCER CC1E LL_TIM_CC_IsEnabledChannel
- CCER CC1NE LL_TIM_CC_IsEnabledChannel
- CCER CC2E LL_TIM_CC_IsEnabledChannel
- CCER CC2NE LL_TIM_CC_IsEnabledChannel
- CCER CC3E LL_TIM_CC_IsEnabledChannel
- CCER CC3NE LL_TIM_CC_IsEnabledChannel
- CCER CC4E LL_TIM_CC_IsEnabledChannel

LL_TIM_OC_ConfigOutput

Function name

__STATIC_INLINE void LL_TIM_OC_ConfigOutput (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Configuration)

Function description

Configure an output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **Configuration:** This parameter must be a combination of all the following values:
  - LL_TIM_OCPOLARITY_HIGH or LL_TIM_OCPOLARITY_LOW
  - LL_TIM_OCIDLESTATE_LOW or LL_TIM_OCIDLESTATE_HIGH

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 CC1S LL_TIM_OC_ConfigOutput
- CCMR1 CC2S LL_TIM_OC_ConfigOutput
- CCMR2 CC3S LL_TIM_OC_ConfigOutput
- CCMR2 CC4S LL_TIM_OC_ConfigOutput
- CCER CC1P LL_TIM_OC_ConfigOutput
- CCER CC2P LL_TIM_OC_ConfigOutput
- CCER CC3P LL_TIM_OC_ConfigOutput
- CCER CC4P LL_TIM_OC_ConfigOutput
- CR2 OIS1 LL_TIM_OC_ConfigOutput
- CR2 OIS2 LL_TIM_OC_ConfigOutput
- CR2 OIS3 LL_TIM_OC_ConfigOutput
- CR2 OIS4 LL_TIM_OC_ConfigOutput

LL_TIM_OC_SetMode

Function name

__STATIC_INLINE void LL_TIM_OC_SetMode (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Mode)

Function description

Define the behavior of the output reference signal OCxREF from which OCx and OCxN (when relevant) are derived.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **Mode:** This parameter can be one of the following values:
  - LL_TIM_OCMODE_FROZEN
  - LL_TIM_OCMODE_ACTIVE
  - LL_TIM_OCMODE_INACTIVE
  - LL_TIM_OCMODE_TOGGLE
  - LL_TIM_OCMODE_FORCED_INACTIVE
  - LL_TIM_OCMODE_FORCED_ACTIVE
  - LL_TIM_OCMODE_PWM1
  - LL_TIM_OCMODE_PWM2

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 OC1M LL_TIM_OC_SetMode
- CCMR1 OC2M LL_TIM_OC_SetMode
- CCMR2 OC3M LL_TIM_OC_SetMode
- CCMR2 OC4M LL_TIM_OC_SetMode

LL_TIM_OC_GetMode

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_GetMode (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Get the output compare mode of an output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_OCMODE_FROZEN
  - LL_TIM_OCMODE_ACTIVE
  - LL_TIM_OCMODE_INACTIVE
  - LL_TIM_OCMODE_TOGGLE
  - LL_TIM_OCMODE_FORCED_INACTIVE
  - LL_TIM_OCMODE_FORCED_ACTIVE
  - LL_TIM_OCMODE_PWM1
  - LL_TIM_OCMODE_PWM2

Reference Manual to LL API cross reference:

- CCMR1 OC1M LL_TIM_OC_GetMode
- CCMR1 OC2M LL_TIM_OC_GetMode
- CCMR2 OC3M LL_TIM_OC_GetMode
- CCMR2 OC4M LL_TIM_OC_GetMode

LL_TIM_OC_SetPolarity

Function name

__STATIC_INLINE void LL_TIM_OC_SetPolarity (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Polarity)

Function description

Set the polarity of an output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH1N
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH2N
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH3N
  - LL_TIM_CHANNEL_CH4
- **Polarity:** This parameter can be one of the following values:
  - LL_TIM_OCPOLARITY_HIGH
  - LL_TIM_OCPOLARITY_LOW

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCER CC1P LL_TIM_OC_SetPolarity
- CCER CC1NP LL_TIM_OC_SetPolarity
- CCER CC2P LL_TIM_OC_SetPolarity
- CCER CC2NP LL_TIM_OC_SetPolarity
- CCER CC3P LL_TIM_OC_SetPolarity
- CCER CC3NP LL_TIM_OC_SetPolarity
- CCER CC4P LL_TIM_OC_SetPolarity

LL_TIM_OC_GetPolarity

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_GetPolarity (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Get the polarity of an output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH1N
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH2N
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH3N
  - LL_TIM_CHANNEL_CH4

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_OCPOLARITY_HIGH
  - LL_TIM_OCPOLARITY_LOW

Reference Manual to LL API cross reference:

- CCER CC1P LL_TIM_OC_GetPolarity
- CCER CC1NP LL_TIM_OC_GetPolarity
- CCER CC2P LL_TIM_OC_GetPolarity
- CCER CC2NP LL_TIM_OC_GetPolarity
- CCER CC3P LL_TIM_OC_GetPolarity
- CCER CC3NP LL_TIM_OC_GetPolarity
- CCER CC4P LL_TIM_OC_GetPolarity

LL_TIM_OC_SetIdleState

Function name

__STATIC_INLINE void LL_TIM_OC_SetIdleState (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t IdleState)

Function description

Set the IDLE state of an output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH1N
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH2N
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH3N
  - LL_TIM_CHANNEL_CH4
- **IdleState:** This parameter can be one of the following values:
  - LL_TIM_OCIDLESTATE_LOW
  - LL_TIM_OCIDLESTATE_HIGH

Return values

- **None:**

Notes

- This function is significant only for the timer instances supporting the break feature. Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- CR2 OIS1 LL_TIM_OC_SetIdleState
- CR2 OIS1N LL_TIM_OC_SetIdleState
- CR2 OIS2 LL_TIM_OC_SetIdleState
- CR2 OIS2N LL_TIM_OC_SetIdleState
- CR2 OIS3 LL_TIM_OC_SetIdleState
- CR2 OIS3N LL_TIM_OC_SetIdleState
- CR2 OIS4 LL_TIM_OC_SetIdleState

LL_TIM_OC_GetIdleState

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_GetIdleState (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Get the IDLE state of an output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH1N
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH2N
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH3N
  - LL_TIM_CHANNEL_CH4

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_OCIDLESTATE_LOW
  - LL_TIM_OCIDLESTATE_HIGH

Reference Manual to LL API cross reference:

- CR2 OIS1 LL_TIM_OC_GetIdleState
- CR2 OIS1N LL_TIM_OC_GetIdleState
- CR2 OIS2 LL_TIM_OC_GetIdleState
- CR2 OIS2N LL_TIM_OC_GetIdleState
- CR2 OIS3 LL_TIM_OC_GetIdleState
- CR2 OIS3N LL_TIM_OC_GetIdleState
- CR2 OIS4 LL_TIM_OC_GetIdleState

LL_TIM_OC_EnableFast

Function name

__STATIC_INLINE void LL_TIM_OC_EnableFast (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Enable fast mode for the output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **None:**

Notes

- Acts only if the channel is configured in PWM1 or PWM2 mode.

Reference Manual to LL API cross reference:

- CCMR1 OC1FE LL_TIM_OC_EnableFast
- CCMR1 OC2FE LL_TIM_OC_EnableFast
- CCMR2 OC3FE LL_TIM_OC_EnableFast
- CCMR2 OC4FE LL_TIM_OC_EnableFast

LL_TIM_OC_DisableFast

Function name

__STATIC_INLINE void LL_TIM_OC_DisableFast (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Disable fast mode for the output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 OC1FE LL_TIM_OC_DisableFast
- CCMR1 OC2FE LL_TIM_OC_DisableFast
- CCMR2 OC3FE LL_TIM_OC_DisableFast
- CCMR2 OC4FE LL_TIM_OC_DisableFast

LL_TIM_OC_IsEnabledFast

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledFast (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Indicates whether fast mode is enabled for the output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CCMR1 OC1FE LL_TIM_OC_IsEnabledFast
- CCMR1 OC2FE LL_TIM_OC_IsEnabledFast
- CCMR2 OC3FE LL_TIM_OC_IsEnabledFast
- CCMR2 OC4FE LL_TIM_OC_IsEnabledFast
-

LL_TIM_OC_EnablePreload

Function name

__STATIC_INLINE void LL_TIM_OC_EnablePreload (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Enable compare register (TIMx_CCRx) preload for the output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 OC1PE LL_TIM_OC_EnablePreload
- CCMR1 OC2PE LL_TIM_OC_EnablePreload
- CCMR2 OC3PE LL_TIM_OC_EnablePreload
- CCMR2 OC4PE LL_TIM_OC_EnablePreload

LL_TIM_OC_DisablePreload

Function name

__STATIC_INLINE void LL_TIM_OC_DisablePreload (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Disable compare register (TIMx_CCRx) preload for the output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 OC1PE LL_TIM_OC_DisablePreload
- CCMR1 OC2PE LL_TIM_OC_DisablePreload
- CCMR2 OC3PE LL_TIM_OC_DisablePreload
- CCMR2 OC4PE LL_TIM_OC_DisablePreload

LL_TIM_OC_IsEnabledPreload

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledPreload (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Indicates whether compare register (TIMx_CCRx) preload is enabled for the output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CCMR1 OC1PE LL_TIM_OC_IsEnabledPreload
- CCMR1 OC2PE LL_TIM_OC_IsEnabledPreload
- CCMR2 OC3PE LL_TIM_OC_IsEnabledPreload
- CCMR2 OC4PE LL_TIM_OC_IsEnabledPreload
-

LL_TIM_OC_EnableClear

Function name

__STATIC_INLINE void LL_TIM_OC_EnableClear (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Enable clearing the output channel on an external event.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **None:**

Notes

- This function can only be used in Output compare and PWM modes. It does not work in Forced mode.
- Macro IS_TIM_OCXREF_CLEAR_INSTANCE(TIMx) can be used to check whether or not a timer instance can clear the OCxREF signal on an external event.

Reference Manual to LL API cross reference:

- CCMR1 OC1CE LL_TIM_OC_EnableClear
- CCMR1 OC2CE LL_TIM_OC_EnableClear
- CCMR2 OC3CE LL_TIM_OC_EnableClear
- CCMR2 OC4CE LL_TIM_OC_EnableClear

LL_TIM_OC_DisableClear

Function name

__STATIC_INLINE void LL_TIM_OC_DisableClear (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Disable clearing the output channel on an external event.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **None:**

Notes

- Macro IS_TIM_OCXREF_CLEAR_INSTANCE(TIMx) can be used to check whether or not a timer instance can clear the OCxREF signal on an external event.

Reference Manual to LL API cross reference:

- CCMR1 OC1CE LL_TIM_OC_DisableClear
- CCMR1 OC2CE LL_TIM_OC_DisableClear
- CCMR2 OC3CE LL_TIM_OC_DisableClear
- CCMR2 OC4CE LL_TIM_OC_DisableClear

LL_TIM_OC_IsEnabledClear

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_IsEnabledClear (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Indicates clearing the output channel on an external event is enabled for the output channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **State:** of bit (1 or 0).

Notes

- This function enables clearing the output channel on an external event.
- This function can only be used in Output compare and PWM modes. It does not work in Forced mode.
- Macro IS_TIM_OCXREF_CLEAR_INSTANCE(TIMx) can be used to check whether or not a timer instance can clear the OCxREF signal on an external event.

Reference Manual to LL API cross reference:

- CCMR1 OC1CE LL_TIM_OC_IsEnabledClear
- CCMR1 OC2CE LL_TIM_OC_IsEnabledClear
- CCMR2 OC3CE LL_TIM_OC_IsEnabledClear
- CCMR2 OC4CE LL_TIM_OC_IsEnabledClear
-

LL_TIM_OC_SetDeadTime

Function name

__STATIC_INLINE void LL_TIM_OC_SetDeadTime (TIM_TypeDef * TIMx, uint32_t DeadTime)

Function description

Set the dead-time delay (delay inserted between the rising edge of the OCxREF signal and the rising edge of the Ocx and OCxN signals).

Parameters

- **TIMx:** Timer instance
- **DeadTime:** between Min_Data=0 and Max_Data=255

Return values

- **None:**

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not dead-time insertion feature is supported by a timer instance.
- Helper macro __LL_TIM_CALC_DEADTIME can be used to calculate the DeadTime parameter

Reference Manual to LL API cross reference:

- BDTR DTG LL_TIM_OC_SetDeadTime

LL_TIM_OC_SetCompareCH1

Function name

__STATIC_INLINE void LL_TIM_OC_SetCompareCH1 (TIM_TypeDef * TIMx, uint32_t CompareValue)

Function description

Set compare value for output channel 1 (TIMx_CCR1).

Parameters

- **TIMx:** Timer instance
- **CompareValue:** between Min_Data=0 and Max_Data=65535

Return values

- **None:**

Notes

- In 32-bit timer implementations compare value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC1_INSTANCE(TIMx) can be used to check whether or not output channel 1 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR1 CCR1 LL_TIM_OC_SetCompareCH1

LL_TIM_OC_SetCompareCH2

Function name

__STATIC_INLINE void LL_TIM_OC_SetCompareCH2 (TIM_TypeDef * TIMx, uint32_t CompareValue)

Function description

Set compare value for output channel 2 (TIMx_CCR2).

Parameters

- **TIMx:** Timer instance
- **CompareValue:** between Min_Data=0 and Max_Data=65535

Return values

- **None:**

Notes

- In 32-bit timer implementations compare value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC2_INSTANCE(TIMx) can be used to check whether or not output channel 2 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR2 CCR2 LL_TIM_OC_SetCompareCH2

LL_TIM_OC_SetCompareCH3

Function name

__STATIC_INLINE void LL_TIM_OC_SetCompareCH3 (TIM_TypeDef * TIMx, uint32_t CompareValue)

Function description

Set compare value for output channel 3 (TIMx_CCR3).

Parameters

- **TIMx:** Timer instance
- **CompareValue:** between Min_Data=0 and Max_Data=65535

Return values

- **None:**

Notes

- In 32-bit timer implementations compare value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC3_INSTANCE(TIMx) can be used to check whether or not output channel is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR3 CCR3 LL_TIM_OC_SetCompareCH3

LL_TIM_OC_SetCompareCH4

Function name

__STATIC_INLINE void LL_TIM_OC_SetCompareCH4 (TIM_TypeDef * TIMx, uint32_t CompareValue)

Function description

Set compare value for output channel 4 (TIMx_CCR4).

Parameters

- **TIMx:** Timer instance
- **CompareValue:** between Min_Data=0 and Max_Data=65535

Return values

- **None:**

Notes

- In 32-bit timer implementations compare value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC4_INSTANCE(TIMx) can be used to check whether or not output channel 4 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR4 CCR4 LL_TIM_OC_SetCompareCH4

LL_TIM_OC_GetCompareCH1

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH1 (TIM_TypeDef * TIMx)

Function description

Get compare value (TIMx_CCR1) set for output channel 1.

Parameters

- **TIMx:** Timer instance

Return values

- **CompareValue:** (between Min_Data=0 and Max_Data=65535)

Notes

- In 32-bit timer implementations returned compare value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC1_INSTANCE(TIMx) can be used to check whether or not output channel 1 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR1 CCR1 LL_TIM_OC_GetCompareCH1

LL_TIM_OC_GetCompareCH2

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH2 (TIM_TypeDef * TIMx)

Function description

Get compare value (TIMx_CCR2) set for output channel 2.

Parameters

- **TIMx:** Timer instance

Return values

- **CompareValue:** (between Min_Data=0 and Max_Data=65535)

Notes

- In 32-bit timer implementations returned compare value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC2_INSTANCE(TIMx) can be used to check whether or not output channel 2 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR2 CCR2 LL_TIM_OC_GetCompareCH2

LL_TIM_OC_GetCompareCH3

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH3 (TIM_TypeDef * TIMx)

Function description

Get compare value (TIMx_CCR3) set for output channel 3.

Parameters

- **TIMx:** Timer instance

Return values

- **CompareValue:** (between Min_Data=0 and Max_Data=65535)

Notes

- In 32-bit timer implementations returned compare value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC3_INSTANCE(TIMx) can be used to check whether or not output channel 3 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR3 CCR3 LL_TIM_OC_GetCompareCH3

LL_TIM_OC_GetCompareCH4

Function name

__STATIC_INLINE uint32_t LL_TIM_OC_GetCompareCH4 (TIM_TypeDef * TIMx)

Function description

Get compare value (TIMx_CCR4) set for output channel 4.

Parameters

- **TIMx:** Timer instance

Return values

- **CompareValue:** (between Min_Data=0 and Max_Data=65535)

Notes

- In 32-bit timer implementations returned compare value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC4_INSTANCE(TIMx) can be used to check whether or not output channel 4 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR4 CCR4 LL_TIM_OC_GetCompareCH4

LL_TIM_IC_Config

Function name

__STATIC_INLINE void LL_TIM_IC_Config (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Configuration)

Function description

Configure input channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **Configuration:** This parameter must be a combination of all the following values:
  - LL_TIM_ACTIVEINPUT_DIRECTTI or LL_TIM_ACTIVEINPUT_INDIRECTTI or LL_TIM_ACTIVEINPUT_TRC
  - LL_TIM_ICPSC_DIV1 or ... or LL_TIM_ICPSC_DIV8
  - LL_TIM_IC_FILTER_FDIV1 or ... or LL_TIM_IC_FILTER_FDIV32_N8
  - LL_TIM_IC_POLARITY_RISING or LL_TIM_IC_POLARITY_FALLING or LL_TIM_IC_POLARITY_BOTHEDGE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 CC1S LL_TIM_IC_Config
- CCMR1 IC1PSC LL_TIM_IC_Config
- CCMR1 IC1F LL_TIM_IC_Config
- CCMR1 CC2S LL_TIM_IC_Config
- CCMR1 IC2PSC LL_TIM_IC_Config
- CCMR1 IC2F LL_TIM_IC_Config
- CCMR2 CC3S LL_TIM_IC_Config
- CCMR2 IC3PSC LL_TIM_IC_Config
- CCMR2 IC3F LL_TIM_IC_Config
- CCMR2 CC4S LL_TIM_IC_Config
- CCMR2 IC4PSC LL_TIM_IC_Config
- CCMR2 IC4F LL_TIM_IC_Config
- CCER CC1P LL_TIM_IC_Config
- CCER CC1NP LL_TIM_IC_Config
- CCER CC2P LL_TIM_IC_Config
- CCER CC2NP LL_TIM_IC_Config
- CCER CC3P LL_TIM_IC_Config
- CCER CC3NP LL_TIM_IC_Config
- CCER CC4P LL_TIM_IC_Config
- CCER CC4NP LL_TIM_IC_Config

LL_TIM_IC_SetActiveInput

Function name

__STATIC_INLINE void LL_TIM_IC_SetActiveInput (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICActiveInput)

Function description

Set the active input.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **ICActiveInput:** This parameter can be one of the following values:
  - LL_TIM_ACTIVEINPUT_DIRECTTI
  - LL_TIM_ACTIVEINPUT_INDIRECTTI
  - LL_TIM_ACTIVEINPUT_TRC

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 CC1S LL_TIM_IC_SetActiveInput
- CCMR1 CC2S LL_TIM_IC_SetActiveInput
- CCMR2 CC3S LL_TIM_IC_SetActiveInput
- CCMR2 CC4S LL_TIM_IC_SetActiveInput

LL_TIM_IC_GetActiveInput

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_GetActiveInput (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Get the current active input.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_ACTIVEINPUT_DIRECTTI
  - LL_TIM_ACTIVEINPUT_INDIRECTTI
  - LL_TIM_ACTIVEINPUT_TRC

Reference Manual to LL API cross reference:

- CCMR1 CC1S LL_TIM_IC_GetActiveInput
- CCMR1 CC2S LL_TIM_IC_GetActiveInput
- CCMR2 CC3S LL_TIM_IC_GetActiveInput
- CCMR2 CC4S LL_TIM_IC_GetActiveInput

LL_TIM_IC_SetPrescaler

Function name

__STATIC_INLINE void LL_TIM_IC_SetPrescaler (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICPrescaler)

Function description

Set the prescaler of input channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **ICPrescaler:** This parameter can be one of the following values:
  - LL_TIM_ICPSC_DIV1
  - LL_TIM_ICPSC_DIV2
  - LL_TIM_ICPSC_DIV4
  - LL_TIM_ICPSC_DIV8

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 IC1PSC LL_TIM_IC_SetPrescaler
- CCMR1 IC2PSC LL_TIM_IC_SetPrescaler
- CCMR2 IC3PSC LL_TIM_IC_SetPrescaler
- CCMR2 IC4PSC LL_TIM_IC_SetPrescaler

LL_TIM_IC_GetPrescaler

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_GetPrescaler (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Get the current prescaler value acting on an input channel.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_ICPSC_DIV1
  - LL_TIM_ICPSC_DIV2
  - LL_TIM_ICPSC_DIV4
  - LL_TIM_ICPSC_DIV8

Reference Manual to LL API cross reference:

- CCMR1 IC1PSC LL_TIM_IC_GetPrescaler
- CCMR1 IC2PSC LL_TIM_IC_GetPrescaler
- CCMR2 IC3PSC LL_TIM_IC_GetPrescaler
- CCMR2 IC4PSC LL_TIM_IC_GetPrescaler

LL_TIM_IC_SetFilter

Function name

__STATIC_INLINE void LL_TIM_IC_SetFilter (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICFilter)

Function description

Set the input filter duration.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **ICFilter:** This parameter can be one of the following values:
  - LL_TIM_IC_FILTER_FDIV1
  - LL_TIM_IC_FILTER_FDIV1_N2
  - LL_TIM_IC_FILTER_FDIV1_N4
  - LL_TIM_IC_FILTER_FDIV1_N8
  - LL_TIM_IC_FILTER_FDIV2_N6
  - LL_TIM_IC_FILTER_FDIV2_N8
  - LL_TIM_IC_FILTER_FDIV4_N6
  - LL_TIM_IC_FILTER_FDIV4_N8
  - LL_TIM_IC_FILTER_FDIV8_N6
  - LL_TIM_IC_FILTER_FDIV8_N8
  - LL_TIM_IC_FILTER_FDIV16_N5
  - LL_TIM_IC_FILTER_FDIV16_N6
  - LL_TIM_IC_FILTER_FDIV16_N8
  - LL_TIM_IC_FILTER_FDIV32_N5
  - LL_TIM_IC_FILTER_FDIV32_N6
  - LL_TIM_IC_FILTER_FDIV32_N8

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCMR1 IC1F LL_TIM_IC_SetFilter
- CCMR1 IC2F LL_TIM_IC_SetFilter
- CCMR2 IC3F LL_TIM_IC_SetFilter
- CCMR2 IC4F LL_TIM_IC_SetFilter

LL_TIM_IC_GetFilter

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_GetFilter (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Get the input filter duration.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_IC_FILTER_FDIV1
  - LL_TIM_IC_FILTER_FDIV1_N2
  - LL_TIM_IC_FILTER_FDIV1_N4
  - LL_TIM_IC_FILTER_FDIV1_N8
  - LL_TIM_IC_FILTER_FDIV2_N6
  - LL_TIM_IC_FILTER_FDIV2_N8
  - LL_TIM_IC_FILTER_FDIV4_N6
  - LL_TIM_IC_FILTER_FDIV4_N8
  - LL_TIM_IC_FILTER_FDIV8_N6
  - LL_TIM_IC_FILTER_FDIV8_N8
  - LL_TIM_IC_FILTER_FDIV16_N5
  - LL_TIM_IC_FILTER_FDIV16_N6
  - LL_TIM_IC_FILTER_FDIV16_N8
  - LL_TIM_IC_FILTER_FDIV32_N5
  - LL_TIM_IC_FILTER_FDIV32_N6
  - LL_TIM_IC_FILTER_FDIV32_N8

Reference Manual to LL API cross reference:

- CCMR1 IC1F LL_TIM_IC_GetFilter
- CCMR1 IC2F LL_TIM_IC_GetFilter
- CCMR2 IC3F LL_TIM_IC_GetFilter
- CCMR2 IC4F LL_TIM_IC_GetFilter

LL_TIM_IC_SetPolarity

Function name

__STATIC_INLINE void LL_TIM_IC_SetPolarity (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICPolarity)

Function description

Set the input channel polarity.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **ICPolarity:** This parameter can be one of the following values:
  - LL_TIM_IC_POLARITY_RISING
  - LL_TIM_IC_POLARITY_FALLING
  - LL_TIM_IC_POLARITY_BOTHEDGE

Return values

- **None:**

Reference Manual to LL API cross reference:

- CCER CC1P LL_TIM_IC_SetPolarity
- CCER CC1NP LL_TIM_IC_SetPolarity
- CCER CC2P LL_TIM_IC_SetPolarity
- CCER CC2NP LL_TIM_IC_SetPolarity
- CCER CC3P LL_TIM_IC_SetPolarity
- CCER CC3NP LL_TIM_IC_SetPolarity
- CCER CC4P LL_TIM_IC_SetPolarity
- CCER CC4NP LL_TIM_IC_SetPolarity

LL_TIM_IC_GetPolarity

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_GetPolarity (TIM_TypeDef * TIMx, uint32_t Channel)

Function description

Get the current input channel polarity.

Parameters

- **TIMx:** Timer instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4

Return values

- **Returned:** value can be one of the following values:
  - LL_TIM_IC_POLARITY_RISING
  - LL_TIM_IC_POLARITY_FALLING
  - LL_TIM_IC_POLARITY_BOTHEDGE

Reference Manual to LL API cross reference:

- CCER CC1P LL_TIM_IC_GetPolarity
- CCER CC1NP LL_TIM_IC_GetPolarity
- CCER CC2P LL_TIM_IC_GetPolarity
- CCER CC2NP LL_TIM_IC_GetPolarity
- CCER CC3P LL_TIM_IC_GetPolarity
- CCER CC3NP LL_TIM_IC_GetPolarity
- CCER CC4P LL_TIM_IC_GetPolarity
- CCER CC4NP LL_TIM_IC_GetPolarity

LL_TIM_IC_EnableXORCombination

Function name

__STATIC_INLINE void LL_TIM_IC_EnableXORCombination (TIM_TypeDef * TIMx)

Function description

Connect the TIMx_CH1, CH2 and CH3 pins to the TI1 input (XOR combination).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_XOR_INSTANCE(TIMx) can be used to check whether or not a timer instance provides an XOR input.

Reference Manual to LL API cross reference:

- CR2 TI1S LL_TIM_IC_EnableXORCombination

LL_TIM_IC_DisableXORCombination

Function name

__STATIC_INLINE void LL_TIM_IC_DisableXORCombination (TIM_TypeDef * TIMx)

Function description

Disconnect the TIMx_CH1, CH2 and CH3 pins from the TI1 input.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_XOR_INSTANCE(TIMx) can be used to check whether or not a timer instance provides an XOR input.

Reference Manual to LL API cross reference:

- CR2 TI1S LL_TIM_IC_DisableXORCombination

LL_TIM_IC_IsEnabledXORCombination

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_IsEnabledXORCombination (TIM_TypeDef * TIMx)

Function description

Indicates whether the TIMx_CH1, CH2 and CH3 pins are connectected to the TI1 input.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_TIM_XOR_INSTANCE(TIMx) can be used to check whether or not a timer instance provides an XOR input.

Reference Manual to LL API cross reference:

- CR2 TI1S LL_TIM_IC_IsEnabledXORCombination

LL_TIM_IC_GetCaptureCH1

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH1 (TIM_TypeDef * TIMx)

Function description

Get captured value for input channel 1.

Parameters

- **TIMx:** Timer instance

Return values

- **CapturedValue:** (between Min_Data=0 and Max_Data=65535)

Notes

- In 32-bit timer implementations returned captured value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC1_INSTANCE(TIMx) can be used to check whether or not input channel 1 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR1 CCR1 LL_TIM_IC_GetCaptureCH1

LL_TIM_IC_GetCaptureCH2

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH2 (TIM_TypeDef * TIMx)

Function description

Get captured value for input channel 2.

Parameters

- **TIMx:** Timer instance

Return values

- **CapturedValue:** (between Min_Data=0 and Max_Data=65535)

Notes

- In 32-bit timer implementations returned captured value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC2_INSTANCE(TIMx) can be used to check whether or not input channel 2 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR2 CCR2 LL_TIM_IC_GetCaptureCH2

LL_TIM_IC_GetCaptureCH3

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH3 (TIM_TypeDef * TIMx)

Function description

Get captured value for input channel 3.

Parameters

- **TIMx:** Timer instance

Return values

- **CapturedValue:** (between Min_Data=0 and Max_Data=65535)

Notes

- In 32-bit timer implementations returned captured value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC3_INSTANCE(TIMx) can be used to check whether or not input channel 3 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR3 CCR3 LL_TIM_IC_GetCaptureCH3

LL_TIM_IC_GetCaptureCH4

Function name

__STATIC_INLINE uint32_t LL_TIM_IC_GetCaptureCH4 (TIM_TypeDef * TIMx)

Function description

Get captured value for input channel 4.

Parameters

- **TIMx:** Timer instance

Return values

- **CapturedValue:** (between Min_Data=0 and Max_Data=65535)

Notes

- In 32-bit timer implementations returned captured value can be between 0x00000000 and 0xFFFFFFFF.
- Macro IS_TIM_32B_COUNTER_INSTANCE(TIMx) can be used to check whether or not a timer instance supports a 32 bits counter.
- Macro IS_TIM_CC4_INSTANCE(TIMx) can be used to check whether or not input channel 4 is supported by a timer instance.

Reference Manual to LL API cross reference:

- CCR4 CCR4 LL_TIM_IC_GetCaptureCH4

LL_TIM_EnableExternalClock

Function name

__STATIC_INLINE void LL_TIM_EnableExternalClock (TIM_TypeDef * TIMx)

Function description

Enable external clock mode 2.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- When external clock mode 2 is enabled the counter is clocked by any active edge on the ETRF signal.
- Macro IS_TIM_CLOCKSOURCE_ETRMODE2_INSTANCE(TIMx) can be used to check whether or not a timer instance supports external clock mode2.

Reference Manual to LL API cross reference:

- SMCR ECE LL_TIM_EnableExternalClock

LL_TIM_DisableExternalClock

Function name

__STATIC_INLINE void LL_TIM_DisableExternalClock (TIM_TypeDef * TIMx)

Function description

Disable external clock mode 2.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_CLOCKSOURCE_ETRMODE2_INSTANCE(TIMx) can be used to check whether or not a timer instance supports external clock mode2.

Reference Manual to LL API cross reference:

- SMCR ECE LL_TIM_DisableExternalClock

LL_TIM_IsEnabledExternalClock

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledExternalClock (TIM_TypeDef * TIMx)

Function description

Indicate whether external clock mode 2 is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_TIM_CLOCKSOURCE_ETRMODE2_INSTANCE(TIMx) can be used to check whether or not a timer instance supports external clock mode2.

Reference Manual to LL API cross reference:

- SMCR ECE LL_TIM_IsEnabledExternalClock

LL_TIM_SetClockSource

Function name

__STATIC_INLINE void LL_TIM_SetClockSource (TIM_TypeDef * TIMx, uint32_t ClockSource)

Function description

Set the clock source of the counter clock.

Parameters

- **TIMx:** Timer instance
- **ClockSource:** This parameter can be one of the following values:
  - LL_TIM_CLOCKSOURCE_INTERNAL
  - LL_TIM_CLOCKSOURCE_EXT_MODE1
  - LL_TIM_CLOCKSOURCE_EXT_MODE2

Return values

- **None:**

Notes

- when selected clock source is external clock mode 1, the timer input the external clock is applied is selected by calling the LL_TIM_SetTriggerInput() function. This timer input must be configured by calling the LL_TIM_IC_Config() function.
- Macro IS_TIM_CLOCKSOURCE_ETRMODE1_INSTANCE(TIMx) can be used to check whether or not a timer instance supports external clock mode1.
- Macro IS_TIM_CLOCKSOURCE_ETRMODE2_INSTANCE(TIMx) can be used to check whether or not a timer instance supports external clock mode2.

Reference Manual to LL API cross reference:

- SMCR SMS LL_TIM_SetClockSource
- SMCR ECE LL_TIM_SetClockSource

LL_TIM_SetEncoderMode

Function name

__STATIC_INLINE void LL_TIM_SetEncoderMode (TIM_TypeDef * TIMx, uint32_t EncoderMode)

Function description

Set the encoder interface mode.

Parameters

- **TIMx:** Timer instance
- **EncoderMode:** This parameter can be one of the following values:
  - LL_TIM_ENCODERMODE_X2_TI1
  - LL_TIM_ENCODERMODE_X2_TI2
  - LL_TIM_ENCODERMODE_X4_TI12

Return values

- **None:**

Notes

- Macro IS_TIM_ENCODER_INTERFACE_INSTANCE(TIMx) can be used to check whether or not a timer instance supports the encoder mode.

Reference Manual to LL API cross reference:

- SMCR SMS LL_TIM_SetEncoderMode

LL_TIM_SetTriggerOutput

Function name

__STATIC_INLINE void LL_TIM_SetTriggerOutput (TIM_TypeDef * TIMx, uint32_t TimerSynchronization)

Function description

Set the trigger output (TRGO) used for timer synchronization .

Parameters

- **TIMx:** Timer instance
- **TimerSynchronization:** This parameter can be one of the following values:
  - LL_TIM_TRGO_RESET
  - LL_TIM_TRGO_ENABLE
  - LL_TIM_TRGO_UPDATE
  - LL_TIM_TRGO_CC1IF
  - LL_TIM_TRGO_OC1REF
  - LL_TIM_TRGO_OC2REF
  - LL_TIM_TRGO_OC3REF
  - LL_TIM_TRGO_OC4REF

Return values

- **None:**

Notes

- Macro IS_TIM_MASTER_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a master timer.

Reference Manual to LL API cross reference:

- CR2 MMS LL_TIM_SetTriggerOutput

LL_TIM_SetSlaveMode

Function name

__STATIC_INLINE void LL_TIM_SetSlaveMode (TIM_TypeDef * TIMx, uint32_t SlaveMode)

Function description

Set the synchronization mode of a slave timer.

Parameters

- **TIMx:** Timer instance
- **SlaveMode:** This parameter can be one of the following values:
  - LL_TIM_SLAVEMODE_DISABLED
  - LL_TIM_SLAVEMODE_RESET
  - LL_TIM_SLAVEMODE_GATED
  - LL_TIM_SLAVEMODE_TRIGGER

Return values

- **None:**

Notes

- Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.

Reference Manual to LL API cross reference:

- SMCR SMS LL_TIM_SetSlaveMode

LL_TIM_SetTriggerInput

Function name

__STATIC_INLINE void LL_TIM_SetTriggerInput (TIM_TypeDef * TIMx, uint32_t TriggerInput)

Function description

Set the selects the trigger input to be used to synchronize the counter.

Parameters

- **TIMx:** Timer instance
- **TriggerInput:** This parameter can be one of the following values:
  - LL_TIM_TS_ITR0
  - LL_TIM_TS_ITR1
  - LL_TIM_TS_ITR2
  - LL_TIM_TS_ITR3
  - LL_TIM_TS_TI1F_ED
  - LL_TIM_TS_TI1FP1
  - LL_TIM_TS_TI2FP2
  - LL_TIM_TS_ETRF

Return values

- **None:**

Notes

- Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.

Reference Manual to LL API cross reference:

- SMCR TS LL_TIM_SetTriggerInput

LL_TIM_EnableMasterSlaveMode

Function name

__STATIC_INLINE void LL_TIM_EnableMasterSlaveMode (TIM_TypeDef * TIMx)

Function description

Enable the Master/Slave mode.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.

Reference Manual to LL API cross reference:

- SMCR MSM LL_TIM_EnableMasterSlaveMode

LL_TIM_DisableMasterSlaveMode

Function name

__STATIC_INLINE void LL_TIM_DisableMasterSlaveMode (TIM_TypeDef * TIMx)

Function description

Disable the Master/Slave mode.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.

Reference Manual to LL API cross reference:

- SMCR MSM LL_TIM_DisableMasterSlaveMode

LL_TIM_IsEnabledMasterSlaveMode

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledMasterSlaveMode (TIM_TypeDef * TIMx)

Function description

Indicates whether the Master/Slave mode is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_TIM_SLAVE_INSTANCE(TIMx) can be used to check whether or not a timer instance can operate as a slave timer.

Reference Manual to LL API cross reference:

- SMCR MSM LL_TIM_IsEnabledMasterSlaveMode

LL_TIM_ConfigETR

Function name

__STATIC_INLINE void LL_TIM_ConfigETR (TIM_TypeDef * TIMx, uint32_t ETRPolarity, uint32_t ETRPrescaler, uint32_t ETRFilter)

Function description

Configure the external trigger (ETR) input.

Parameters

- **TIMx:** Timer instance
- **ETRPolarity:** This parameter can be one of the following values:
  - LL_TIM_ETR_POLARITY_NONINVERTED
  - LL_TIM_ETR_POLARITY_INVERTED
- **ETRPrescaler:** This parameter can be one of the following values:
  - LL_TIM_ETR_PRESCALER_DIV1
  - LL_TIM_ETR_PRESCALER_DIV2
  - LL_TIM_ETR_PRESCALER_DIV4
  - LL_TIM_ETR_PRESCALER_DIV8
- **ETRFilter:** This parameter can be one of the following values:
  - LL_TIM_ETR_FILTER_FDIV1
  - LL_TIM_ETR_FILTER_FDIV1_N2
  - LL_TIM_ETR_FILTER_FDIV1_N4
  - LL_TIM_ETR_FILTER_FDIV1_N8
  - LL_TIM_ETR_FILTER_FDIV2_N6
  - LL_TIM_ETR_FILTER_FDIV2_N8
  - LL_TIM_ETR_FILTER_FDIV4_N6
  - LL_TIM_ETR_FILTER_FDIV4_N8
  - LL_TIM_ETR_FILTER_FDIV8_N6
  - LL_TIM_ETR_FILTER_FDIV8_N8
  - LL_TIM_ETR_FILTER_FDIV16_N5
  - LL_TIM_ETR_FILTER_FDIV16_N6
  - LL_TIM_ETR_FILTER_FDIV16_N8
  - LL_TIM_ETR_FILTER_FDIV32_N5
  - LL_TIM_ETR_FILTER_FDIV32_N6
  - LL_TIM_ETR_FILTER_FDIV32_N8

Return values

- **None:**

Notes

- Macro IS_TIM_ETR_INSTANCE(TIMx) can be used to check whether or not a timer instance provides an external trigger input.

Reference Manual to LL API cross reference:

- SMCR ETP LL_TIM_ConfigETR
- SMCR ETPS LL_TIM_ConfigETR
- SMCR ETF LL_TIM_ConfigETR

LL_TIM_EnableBRK

Function name

__STATIC_INLINE void LL_TIM_EnableBRK (TIM_TypeDef * TIMx)

Function description

Enable the break function.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR BKE LL_TIM_EnableBRK

LL_TIM_DisableBRK

Function name

__STATIC_INLINE void LL_TIM_DisableBRK (TIM_TypeDef * TIMx)

Function description

Disable the break function.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR BKE LL_TIM_DisableBRK

LL_TIM_ConfigBRK

Function name

__STATIC_INLINE void LL_TIM_ConfigBRK (TIM_TypeDef * TIMx, uint32_t BreakPolarity)

Function description

Configure the break input.

Parameters

- **TIMx:** Timer instance
- **BreakPolarity:** This parameter can be one of the following values:
  - LL_TIM_BREAK_POLARITY_LOW
  - LL_TIM_BREAK_POLARITY_HIGH

Return values

- **None:**

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR BKP LL_TIM_ConfigBRK

LL_TIM_SetOffStates

Function name

__STATIC_INLINE void LL_TIM_SetOffStates (TIM_TypeDef * TIMx, uint32_t OffStateIdle, uint32_t OffStateRun)

Function description

Select the outputs off state (enabled v.s.

Parameters

- **TIMx:** Timer instance
- **OffStateIdle:** This parameter can be one of the following values:
  - LL_TIM_OSSI_DISABLE
  - LL_TIM_OSSI_ENABLE
- **OffStateRun:** This parameter can be one of the following values:
  - LL_TIM_OSSR_DISABLE
  - LL_TIM_OSSR_ENABLE

Return values

- **None:**

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR OSSI LL_TIM_SetOffStates
- BDTR OSSR LL_TIM_SetOffStates

LL_TIM_EnableAutomaticOutput

Function name

__STATIC_INLINE void LL_TIM_EnableAutomaticOutput (TIM_TypeDef * TIMx)

Function description

Enable automatic output (MOE can be set by software or automatically when a break input is active).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR AOE LL_TIM_EnableAutomaticOutput

LL_TIM_DisableAutomaticOutput

Function name

__STATIC_INLINE void LL_TIM_DisableAutomaticOutput (TIM_TypeDef * TIMx)

Function description

Disable automatic output (MOE can be set only by software).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR AOE LL_TIM_DisableAutomaticOutput

LL_TIM_IsEnabledAutomaticOutput

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledAutomaticOutput (TIM_TypeDef * TIMx)

Function description

Indicate whether automatic output is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR AOE LL_TIM_IsEnabledAutomaticOutput

LL_TIM_EnableAllOutputs

Function name

__STATIC_INLINE void LL_TIM_EnableAllOutputs (TIM_TypeDef * TIMx)

Function description

Enable the outputs (set the MOE bit in TIMx_BDTR register).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- The MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event
- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR MOE LL_TIM_EnableAllOutputs

LL_TIM_DisableAllOutputs

Function name

__STATIC_INLINE void LL_TIM_DisableAllOutputs (TIM_TypeDef * TIMx)

Function description

Disable the outputs (reset the MOE bit in TIMx_BDTR register).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Notes

- The MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR MOE LL_TIM_DisableAllOutputs

LL_TIM_IsEnabledAllOutputs

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledAllOutputs (TIM_TypeDef * TIMx)

Function description

Indicates whether outputs are enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

Reference Manual to LL API cross reference:

- BDTR MOE LL_TIM_IsEnabledAllOutputs

LL_TIM_ConfigDMABurst

Function name

__STATIC_INLINE void LL_TIM_ConfigDMABurst (TIM_TypeDef * TIMx, uint32_t DMABurstBaseAddress, uint32_t DMABurstLength)

Function description

Configures the timer DMA burst feature.

Parameters

- **TIMx:** Timer instance
- **DMABurstBaseAddress:** This parameter can be one of the following values:
  - LL_TIM_DMABURST_BASEADDR_CR1
  - LL_TIM_DMABURST_BASEADDR_CR2
  - LL_TIM_DMABURST_BASEADDR_SMCR
  - LL_TIM_DMABURST_BASEADDR_DIER
  - LL_TIM_DMABURST_BASEADDR_SR
  - LL_TIM_DMABURST_BASEADDR_EGR
  - LL_TIM_DMABURST_BASEADDR_CCMR1
  - LL_TIM_DMABURST_BASEADDR_CCMR2
  - LL_TIM_DMABURST_BASEADDR_CCER
  - LL_TIM_DMABURST_BASEADDR_CNT
  - LL_TIM_DMABURST_BASEADDR_PSC
  - LL_TIM_DMABURST_BASEADDR_ARR
  - LL_TIM_DMABURST_BASEADDR_RCR
  - LL_TIM_DMABURST_BASEADDR_CCR1
  - LL_TIM_DMABURST_BASEADDR_CCR2
  - LL_TIM_DMABURST_BASEADDR_CCR3
  - LL_TIM_DMABURST_BASEADDR_CCR4
  - LL_TIM_DMABURST_BASEADDR_BDTR
- **DMABurstLength:** This parameter can be one of the following values:
  - LL_TIM_DMABURST_LENGTH_1TRANSFER
  - LL_TIM_DMABURST_LENGTH_2TRANSFERS
  - LL_TIM_DMABURST_LENGTH_3TRANSFERS
  - LL_TIM_DMABURST_LENGTH_4TRANSFERS
  - LL_TIM_DMABURST_LENGTH_5TRANSFERS
  - LL_TIM_DMABURST_LENGTH_6TRANSFERS
  - LL_TIM_DMABURST_LENGTH_7TRANSFERS
  - LL_TIM_DMABURST_LENGTH_8TRANSFERS
  - LL_TIM_DMABURST_LENGTH_9TRANSFERS
  - LL_TIM_DMABURST_LENGTH_10TRANSFERS
  - LL_TIM_DMABURST_LENGTH_11TRANSFERS
  - LL_TIM_DMABURST_LENGTH_12TRANSFERS
  - LL_TIM_DMABURST_LENGTH_13TRANSFERS
  - LL_TIM_DMABURST_LENGTH_14TRANSFERS
  - LL_TIM_DMABURST_LENGTH_15TRANSFERS
  - LL_TIM_DMABURST_LENGTH_16TRANSFERS
  - LL_TIM_DMABURST_LENGTH_17TRANSFERS
  - LL_TIM_DMABURST_LENGTH_18TRANSFERS

Return values

- **None:**

Notes

- Macro IS_TIM_DMABURST_INSTANCE(TIMx) can be used to check whether or not a timer instance supports the DMA burst mode.

Reference Manual to LL API cross reference:

- DCR DBL LL_TIM_ConfigDMABurst
- DCR DBA LL_TIM_ConfigDMABurst

LL_TIM_SetRemap

Function name

__STATIC_INLINE void LL_TIM_SetRemap (TIM_TypeDef * TIMx, uint32_t Remap)

Function description

Remap TIM inputs (input channel, internal/external triggers).

Parameters

- **TIMx:** Timer instance
- **Remap:** Remap param depends on the TIMx. Description available only in CHM version of the User Manual (not in .pdf). Otherwise see Reference Manual description of OR registers.

Return values

- **None:**

Notes

- Macro IS_TIM_REMAP_INSTANCE(TIMx) can be used to check whether or not a some timer inputs can be remapped.

Reference Manual to LL API cross reference:

- TIM1_OR ITR2_RMP LL_TIM_SetRemap
- TIM2_OR ITR1_RMP LL_TIM_SetRemap
- TIM5_OR ITR1_RMP LL_TIM_SetRemap
- TIM5_OR TI4_RMP LL_TIM_SetRemap
- TIM9_OR ITR1_RMP LL_TIM_SetRemap
- TIM11_OR TI1_RMP LL_TIM_SetRemap
- LPTIM1_OR OR LL_TIM_SetRemap

LL_TIM_ClearFlag_UPDATE

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_UPDATE (TIM_TypeDef * TIMx)

Function description

Clear the update interrupt flag (UIF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR UIF LL_TIM_ClearFlag_UPDATE

LL_TIM_IsActiveFlag_UPDATE

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_UPDATE (TIM_TypeDef * TIMx)

Function description

Indicate whether update interrupt flag (UIF) is set (update interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR UIF LL_TIM_IsActiveFlag_UPDATE

LL_TIM_ClearFlag_CC1

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_CC1 (TIM_TypeDef * TIMx)

Function description

Clear the Capture/Compare 1 interrupt flag (CC1F).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CC1IF LL_TIM_ClearFlag_CC1

LL_TIM_IsActiveFlag_CC1

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC1 (TIM_TypeDef * TIMx)

Function description

Indicate whether Capture/Compare 1 interrupt flag (CC1F) is set (Capture/Compare 1 interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CC1IF LL_TIM_IsActiveFlag_CC1

LL_TIM_ClearFlag_CC2

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_CC2 (TIM_TypeDef * TIMx)

Function description

Clear the Capture/Compare 2 interrupt flag (CC2F).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CC2IF LL_TIM_ClearFlag_CC2

LL_TIM_IsActiveFlag_CC2

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC2 (TIM_TypeDef * TIMx)

Function description

Indicate whether Capture/Compare 2 interrupt flag (CC2F) is set (Capture/Compare 2 interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CC2IF LL_TIM_IsActiveFlag_CC2

LL_TIM_ClearFlag_CC3

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_CC3 (TIM_TypeDef * TIMx)

Function description

Clear the Capture/Compare 3 interrupt flag (CC3F).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CC3IF LL_TIM_ClearFlag_CC3

LL_TIM_IsActiveFlag_CC3

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC3 (TIM_TypeDef * TIMx)

Function description

Indicate whether Capture/Compare 3 interrupt flag (CC3F) is set (Capture/Compare 3 interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CC3IF LL_TIM_IsActiveFlag_CC3

LL_TIM_ClearFlag_CC4

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_CC4 (TIM_TypeDef * TIMx)

Function description

Clear the Capture/Compare 4 interrupt flag (CC4F).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CC4IF LL_TIM_ClearFlag_CC4

LL_TIM_IsActiveFlag_CC4

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC4 (TIM_TypeDef * TIMx)

Function description

Indicate whether Capture/Compare 4 interrupt flag (CC4F) is set (Capture/Compare 4 interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CC4IF LL_TIM_IsActiveFlag_CC4

LL_TIM_ClearFlag_COM

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_COM (TIM_TypeDef * TIMx)

Function description

Clear the commutation interrupt flag (COMIF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR COMIF LL_TIM_ClearFlag_COM

LL_TIM_IsActiveFlag_COM

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_COM (TIM_TypeDef * TIMx)

Function description

Indicate whether commutation interrupt flag (COMIF) is set (commutation interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR COMIF LL_TIM_IsActiveFlag_COM

LL_TIM_ClearFlag_TRIG

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_TRIG (TIM_TypeDef * TIMx)

Function description

Clear the trigger interrupt flag (TIF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR TIF LL_TIM_ClearFlag_TRIG

LL_TIM_IsActiveFlag_TRIG

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_TRIG (TIM_TypeDef * TIMx)

Function description

Indicate whether trigger interrupt flag (TIF) is set (trigger interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR TIF LL_TIM_IsActiveFlag_TRIG

LL_TIM_ClearFlag_BRK

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_BRK (TIM_TypeDef * TIMx)

Function description

Clear the break interrupt flag (BIF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR BIF LL_TIM_ClearFlag_BRK

LL_TIM_IsActiveFlag_BRK

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_BRK (TIM_TypeDef * TIMx)

Function description

Indicate whether break interrupt flag (BIF) is set (break interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR BIF LL_TIM_IsActiveFlag_BRK

LL_TIM_ClearFlag_CC1OVR

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_CC1OVR (TIM_TypeDef * TIMx)

Function description

Clear the Capture/Compare 1 over-capture interrupt flag (CC1OF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CC1OF LL_TIM_ClearFlag_CC1OVR

LL_TIM_IsActiveFlag_CC1OVR

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC1OVR (TIM_TypeDef * TIMx)

Function description

Indicate whether Capture/Compare 1 over-capture interrupt flag (CC1OF) is set (Capture/Compare 1 interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CC1OF LL_TIM_IsActiveFlag_CC1OVR

LL_TIM_ClearFlag_CC2OVR

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_CC2OVR (TIM_TypeDef * TIMx)

Function description

Clear the Capture/Compare 2 over-capture interrupt flag (CC2OF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CC2OF LL_TIM_ClearFlag_CC2OVR

LL_TIM_IsActiveFlag_CC2OVR

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC2OVR (TIM_TypeDef * TIMx)

Function description

Indicate whether Capture/Compare 2 over-capture interrupt flag (CC2OF) is set (Capture/Compare 2 over-capture interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CC2OF LL_TIM_IsActiveFlag_CC2OVR

LL_TIM_ClearFlag_CC3OVR

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_CC3OVR (TIM_TypeDef * TIMx)

Function description

Clear the Capture/Compare 3 over-capture interrupt flag (CC3OF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CC3OF LL_TIM_ClearFlag_CC3OVR

LL_TIM_IsActiveFlag_CC3OVR

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC3OVR (TIM_TypeDef * TIMx)

Function description

Indicate whether Capture/Compare 3 over-capture interrupt flag (CC3OF) is set (Capture/Compare 3 over-capture interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CC3OF LL_TIM_IsActiveFlag_CC3OVR

LL_TIM_ClearFlag_CC4OVR

Function name

__STATIC_INLINE void LL_TIM_ClearFlag_CC4OVR (TIM_TypeDef * TIMx)

Function description

Clear the Capture/Compare 4 over-capture interrupt flag (CC4OF).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR CC4OF LL_TIM_ClearFlag_CC4OVR

LL_TIM_IsActiveFlag_CC4OVR

Function name

__STATIC_INLINE uint32_t LL_TIM_IsActiveFlag_CC4OVR (TIM_TypeDef * TIMx)

Function description

Indicate whether Capture/Compare 4 over-capture interrupt flag (CC4OF) is set (Capture/Compare 4 over-capture interrupt is pending).

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR CC4OF LL_TIM_IsActiveFlag_CC4OVR

LL_TIM_EnableIT_UPDATE

Function name

__STATIC_INLINE void LL_TIM_EnableIT_UPDATE (TIM_TypeDef * TIMx)

Function description

Enable update interrupt (UIE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER UIE LL_TIM_EnableIT_UPDATE

LL_TIM_DisableIT_UPDATE

Function name

__STATIC_INLINE void LL_TIM_DisableIT_UPDATE (TIM_TypeDef * TIMx)

Function description

Disable update interrupt (UIE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER UIE LL_TIM_DisableIT_UPDATE

LL_TIM_IsEnabledIT_UPDATE

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_UPDATE (TIM_TypeDef * TIMx)

Function description

Indicates whether the update interrupt (UIE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER UIE LL_TIM_IsEnabledIT_UPDATE

LL_TIM_EnableIT_CC1

Function name

__STATIC_INLINE void LL_TIM_EnableIT_CC1 (TIM_TypeDef * TIMx)

Function description

Enable capture/compare 1 interrupt (CC1IE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC1IE LL_TIM_EnableIT_CC1

LL_TIM_DisableIT_CC1

Function name

__STATIC_INLINE void LL_TIM_DisableIT_CC1 (TIM_TypeDef * TIMx)

Function description

Disable capture/compare 1 interrupt (CC1IE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC1IE LL_TIM_DisableIT_CC1

LL_TIM_IsEnabledIT_CC1

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC1 (TIM_TypeDef * TIMx)

Function description

Indicates whether the capture/compare 1 interrupt (CC1IE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER CC1IE LL_TIM_IsEnabledIT_CC1

LL_TIM_EnableIT_CC2

Function name

__STATIC_INLINE void LL_TIM_EnableIT_CC2 (TIM_TypeDef * TIMx)

Function description

Enable capture/compare 2 interrupt (CC2IE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC2IE LL_TIM_EnableIT_CC2

LL_TIM_DisableIT_CC2

Function name

__STATIC_INLINE void LL_TIM_DisableIT_CC2 (TIM_TypeDef * TIMx)

Function description

Disable capture/compare 2 interrupt (CC2IE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC2IE LL_TIM_DisableIT_CC2

LL_TIM_IsEnabledIT_CC2

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC2 (TIM_TypeDef * TIMx)

Function description

Indicates whether the capture/compare 2 interrupt (CC2IE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER CC2IE LL_TIM_IsEnabledIT_CC2

LL_TIM_EnableIT_CC3

Function name

__STATIC_INLINE void LL_TIM_EnableIT_CC3 (TIM_TypeDef * TIMx)

Function description

Enable capture/compare 3 interrupt (CC3IE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC3IE LL_TIM_EnableIT_CC3

LL_TIM_DisableIT_CC3

Function name

__STATIC_INLINE void LL_TIM_DisableIT_CC3 (TIM_TypeDef * TIMx)

Function description

Disable capture/compare 3 interrupt (CC3IE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC3IE LL_TIM_DisableIT_CC3

LL_TIM_IsEnabledIT_CC3

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC3 (TIM_TypeDef * TIMx)

Function description

Indicates whether the capture/compare 3 interrupt (CC3IE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER CC3IE LL_TIM_IsEnabledIT_CC3

LL_TIM_EnableIT_CC4

Function name

__STATIC_INLINE void LL_TIM_EnableIT_CC4 (TIM_TypeDef * TIMx)

Function description

Enable capture/compare 4 interrupt (CC4IE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC4IE LL_TIM_EnableIT_CC4

LL_TIM_DisableIT_CC4

Function name

__STATIC_INLINE void LL_TIM_DisableIT_CC4 (TIM_TypeDef * TIMx)

Function description

Disable capture/compare 4 interrupt (CC4IE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC4IE LL_TIM_DisableIT_CC4

LL_TIM_IsEnabledIT_CC4

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_CC4 (TIM_TypeDef * TIMx)

Function description

Indicates whether the capture/compare 4 interrupt (CC4IE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER CC4IE LL_TIM_IsEnabledIT_CC4

LL_TIM_EnableIT_COM

Function name

__STATIC_INLINE void LL_TIM_EnableIT_COM (TIM_TypeDef * TIMx)

Function description

Enable commutation interrupt (COMIE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER COMIE LL_TIM_EnableIT_COM

LL_TIM_DisableIT_COM

Function name

__STATIC_INLINE void LL_TIM_DisableIT_COM (TIM_TypeDef * TIMx)

Function description

Disable commutation interrupt (COMIE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER COMIE LL_TIM_DisableIT_COM

LL_TIM_IsEnabledIT_COM

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_COM (TIM_TypeDef * TIMx)

Function description

Indicates whether the commutation interrupt (COMIE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER COMIE LL_TIM_IsEnabledIT_COM

LL_TIM_EnableIT_TRIG

Function name

__STATIC_INLINE void LL_TIM_EnableIT_TRIG (TIM_TypeDef * TIMx)

Function description

Enable trigger interrupt (TIE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER TIE LL_TIM_EnableIT_TRIG

LL_TIM_DisableIT_TRIG

Function name

__STATIC_INLINE void LL_TIM_DisableIT_TRIG (TIM_TypeDef * TIMx)

Function description

Disable trigger interrupt (TIE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER TIE LL_TIM_DisableIT_TRIG

LL_TIM_IsEnabledIT_TRIG

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_TRIG (TIM_TypeDef * TIMx)

Function description

Indicates whether the trigger interrupt (TIE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER TIE LL_TIM_IsEnabledIT_TRIG

LL_TIM_EnableIT_BRK

Function name

__STATIC_INLINE void LL_TIM_EnableIT_BRK (TIM_TypeDef * TIMx)

Function description

Enable break interrupt (BIE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER BIE LL_TIM_EnableIT_BRK

LL_TIM_DisableIT_BRK

Function name

__STATIC_INLINE void LL_TIM_DisableIT_BRK (TIM_TypeDef * TIMx)

Function description

Disable break interrupt (BIE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER BIE LL_TIM_DisableIT_BRK

LL_TIM_IsEnabledIT_BRK

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledIT_BRK (TIM_TypeDef * TIMx)

Function description

Indicates whether the break interrupt (BIE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER BIE LL_TIM_IsEnabledIT_BRK

LL_TIM_EnableDMAReq_UPDATE

Function name

__STATIC_INLINE void LL_TIM_EnableDMAReq_UPDATE (TIM_TypeDef * TIMx)

Function description

Enable update DMA request (UDE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER UDE LL_TIM_EnableDMAReq_UPDATE

LL_TIM_DisableDMAReq_UPDATE

Function name

__STATIC_INLINE void LL_TIM_DisableDMAReq_UPDATE (TIM_TypeDef * TIMx)

Function description

Disable update DMA request (UDE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER UDE LL_TIM_DisableDMAReq_UPDATE

LL_TIM_IsEnabledDMAReq_UPDATE

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_UPDATE (TIM_TypeDef * TIMx)

Function description

Indicates whether the update DMA request (UDE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER UDE LL_TIM_IsEnabledDMAReq_UPDATE

LL_TIM_EnableDMAReq_CC1

Function name

__STATIC_INLINE void LL_TIM_EnableDMAReq_CC1 (TIM_TypeDef * TIMx)

Function description

Enable capture/compare 1 DMA request (CC1DE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC1DE LL_TIM_EnableDMAReq_CC1

LL_TIM_DisableDMAReq_CC1

Function name

__STATIC_INLINE void LL_TIM_DisableDMAReq_CC1 (TIM_TypeDef * TIMx)

Function description

Disable capture/compare 1 DMA request (CC1DE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC1DE LL_TIM_DisableDMAReq_CC1

LL_TIM_IsEnabledDMAReq_CC1

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC1 (TIM_TypeDef * TIMx)

Function description

Indicates whether the capture/compare 1 DMA request (CC1DE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER CC1DE LL_TIM_IsEnabledDMAReq_CC1

LL_TIM_EnableDMAReq_CC2

Function name

__STATIC_INLINE void LL_TIM_EnableDMAReq_CC2 (TIM_TypeDef * TIMx)

Function description

Enable capture/compare 2 DMA request (CC2DE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC2DE LL_TIM_EnableDMAReq_CC2

LL_TIM_DisableDMAReq_CC2

Function name

__STATIC_INLINE void LL_TIM_DisableDMAReq_CC2 (TIM_TypeDef * TIMx)

Function description

Disable capture/compare 2 DMA request (CC2DE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC2DE LL_TIM_DisableDMAReq_CC2

LL_TIM_IsEnabledDMAReq_CC2

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC2 (TIM_TypeDef * TIMx)

Function description

Indicates whether the capture/compare 2 DMA request (CC2DE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER CC2DE LL_TIM_IsEnabledDMAReq_CC2

LL_TIM_EnableDMAReq_CC3

Function name

__STATIC_INLINE void LL_TIM_EnableDMAReq_CC3 (TIM_TypeDef * TIMx)

Function description

Enable capture/compare 3 DMA request (CC3DE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC3DE LL_TIM_EnableDMAReq_CC3

LL_TIM_DisableDMAReq_CC3

Function name

__STATIC_INLINE void LL_TIM_DisableDMAReq_CC3 (TIM_TypeDef * TIMx)

Function description

Disable capture/compare 3 DMA request (CC3DE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC3DE LL_TIM_DisableDMAReq_CC3

LL_TIM_IsEnabledDMAReq_CC3

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC3 (TIM_TypeDef * TIMx)

Function description

Indicates whether the capture/compare 3 DMA request (CC3DE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER CC3DE LL_TIM_IsEnabledDMAReq_CC3

LL_TIM_EnableDMAReq_CC4

Function name

__STATIC_INLINE void LL_TIM_EnableDMAReq_CC4 (TIM_TypeDef * TIMx)

Function description

Enable capture/compare 4 DMA request (CC4DE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC4DE LL_TIM_EnableDMAReq_CC4

LL_TIM_DisableDMAReq_CC4

Function name

__STATIC_INLINE void LL_TIM_DisableDMAReq_CC4 (TIM_TypeDef * TIMx)

Function description

Disable capture/compare 4 DMA request (CC4DE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER CC4DE LL_TIM_DisableDMAReq_CC4

LL_TIM_IsEnabledDMAReq_CC4

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_CC4 (TIM_TypeDef * TIMx)

Function description

Indicates whether the capture/compare 4 DMA request (CC4DE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER CC4DE LL_TIM_IsEnabledDMAReq_CC4

LL_TIM_EnableDMAReq_COM

Function name

__STATIC_INLINE void LL_TIM_EnableDMAReq_COM (TIM_TypeDef * TIMx)

Function description

Enable commutation DMA request (COMDE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER COMDE LL_TIM_EnableDMAReq_COM

LL_TIM_DisableDMAReq_COM

Function name

__STATIC_INLINE void LL_TIM_DisableDMAReq_COM (TIM_TypeDef * TIMx)

Function description

Disable commutation DMA request (COMDE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER COMDE LL_TIM_DisableDMAReq_COM

LL_TIM_IsEnabledDMAReq_COM

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_COM (TIM_TypeDef * TIMx)

Function description

Indicates whether the commutation DMA request (COMDE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER COMDE LL_TIM_IsEnabledDMAReq_COM

LL_TIM_EnableDMAReq_TRIG

Function name

__STATIC_INLINE void LL_TIM_EnableDMAReq_TRIG (TIM_TypeDef * TIMx)

Function description

Enable trigger interrupt (TDE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER TDE LL_TIM_EnableDMAReq_TRIG

LL_TIM_DisableDMAReq_TRIG

Function name

__STATIC_INLINE void LL_TIM_DisableDMAReq_TRIG (TIM_TypeDef * TIMx)

Function description

Disable trigger interrupt (TDE).

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- DIER TDE LL_TIM_DisableDMAReq_TRIG

LL_TIM_IsEnabledDMAReq_TRIG

Function name

__STATIC_INLINE uint32_t LL_TIM_IsEnabledDMAReq_TRIG (TIM_TypeDef * TIMx)

Function description

Indicates whether the trigger interrupt (TDE) is enabled.

Parameters

- **TIMx:** Timer instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- DIER TDE LL_TIM_IsEnabledDMAReq_TRIG

LL_TIM_GenerateEvent_UPDATE

Function name

__STATIC_INLINE void LL_TIM_GenerateEvent_UPDATE (TIM_TypeDef * TIMx)

Function description

Generate an update event.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- EGR UG LL_TIM_GenerateEvent_UPDATE

LL_TIM_GenerateEvent_CC1

Function name

__STATIC_INLINE void LL_TIM_GenerateEvent_CC1 (TIM_TypeDef * TIMx)

Function description

Generate Capture/Compare 1 event.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- EGR CC1G LL_TIM_GenerateEvent_CC1

LL_TIM_GenerateEvent_CC2

Function name

__STATIC_INLINE void LL_TIM_GenerateEvent_CC2 (TIM_TypeDef * TIMx)

Function description

Generate Capture/Compare 2 event.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- EGR CC2G LL_TIM_GenerateEvent_CC2

LL_TIM_GenerateEvent_CC3

Function name

__STATIC_INLINE void LL_TIM_GenerateEvent_CC3 (TIM_TypeDef * TIMx)

Function description

Generate Capture/Compare 3 event.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- EGR CC3G LL_TIM_GenerateEvent_CC3

LL_TIM_GenerateEvent_CC4

Function name

__STATIC_INLINE void LL_TIM_GenerateEvent_CC4 (TIM_TypeDef * TIMx)

Function description

Generate Capture/Compare 4 event.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- EGR CC4G LL_TIM_GenerateEvent_CC4

LL_TIM_GenerateEvent_COM

Function name

__STATIC_INLINE void LL_TIM_GenerateEvent_COM (TIM_TypeDef * TIMx)

Function description

Generate commutation event.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- EGR COMG LL_TIM_GenerateEvent_COM

LL_TIM_GenerateEvent_TRIG

Function name

__STATIC_INLINE void LL_TIM_GenerateEvent_TRIG (TIM_TypeDef * TIMx)

Function description

Generate trigger event.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- EGR TG LL_TIM_GenerateEvent_TRIG

LL_TIM_GenerateEvent_BRK

Function name

__STATIC_INLINE void LL_TIM_GenerateEvent_BRK (TIM_TypeDef * TIMx)

Function description

Generate break event.

Parameters

- **TIMx:** Timer instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- EGR BG LL_TIM_GenerateEvent_BRK

LL_TIM_DeInit

Function name

ErrorStatus LL_TIM_DeInit (TIM_TypeDef * TIMx)

Function description

Set TIMx registers to their reset values.

Parameters

- **TIMx:** Timer instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: TIMx registers are de-initialized
  - ERROR: invalid TIMx instance

LL_TIM_StructInit

Function name

void LL_TIM_StructInit (LL_TIM_InitTypeDef * TIM_InitStruct)

Function description

Set the fields of the time base unit configuration data structure to their default values.

Parameters

- **TIM_InitStruct:** pointer to a LL_TIM_InitTypeDef structure (time base unit configuration data structure)

Return values

- **None:**

LL_TIM_Init

Function name

ErrorStatus LL_TIM_Init (TIM_TypeDef * TIMx, LL_TIM_InitTypeDef * TIM_InitStruct)

Function description

Configure the TIMx time base unit.

Parameters

- **TIMx:** Timer Instance
- **TIM_InitStruct:** pointer to a LL_TIM_InitTypeDef structure (TIMx time base unit configuration data structure)

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: TIMx registers are de-initialized
  - ERROR: not applicable

LL_TIM_OC_StructInit

Function name

void LL_TIM_OC_StructInit (LL_TIM_OC_InitTypeDef * TIM_OC_InitStruct)

Function description

Set the fields of the TIMx output channel configuration data structure to their default values.

Parameters

- **TIM_OC_InitStruct:** pointer to a LL_TIM_OC_InitTypeDef structure (the output channel configuration data structure)

Return values

- **None:**

LL_TIM_OC_Init

Function name

ErrorStatus LL_TIM_OC_Init (TIM_TypeDef * TIMx, uint32_t Channel, LL_TIM_OC_InitTypeDef * TIM_OC_InitStruct)

Function description

Configure the TIMx output channel.

Parameters

- **TIMx:** Timer Instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **TIM_OC_InitStruct:** pointer to a LL_TIM_OC_InitTypeDef structure (TIMx output channel configuration data structure)

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: TIMx output channel is initialized
  - ERROR: TIMx output channel is not initialized

LL_TIM_IC_StructInit

Function name

void LL_TIM_IC_StructInit (LL_TIM_IC_InitTypeDef * TIM_ICInitStruct)

Function description

Set the fields of the TIMx input channel configuration data structure to their default values.

Parameters

- **TIM_ICInitStruct:** pointer to a LL_TIM_IC_InitTypeDef structure (the input channel configuration data structure)

Return values

- **None:**

LL_TIM_IC_Init

Function name

ErrorStatus LL_TIM_IC_Init (TIM_TypeDef * TIMx, uint32_t Channel, LL_TIM_IC_InitTypeDef * TIM_IC_InitStruct)

Function description

Configure the TIMx input channel.

Parameters

- **TIMx:** Timer Instance
- **Channel:** This parameter can be one of the following values:
  - LL_TIM_CHANNEL_CH1
  - LL_TIM_CHANNEL_CH2
  - LL_TIM_CHANNEL_CH3
  - LL_TIM_CHANNEL_CH4
- **TIM_IC_InitStruct:** pointer to a LL_TIM_IC_InitTypeDef structure (TIMx input channel configuration data structure)

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: TIMx output channel is initialized
  - ERROR: TIMx output channel is not initialized

LL_TIM_ENCODER_StructInit

Function name

void LL_TIM_ENCODER_StructInit (LL_TIM_ENCODER_InitTypeDef * TIM_EncoderInitStruct)

Function description

Fills each TIM_EncoderInitStruct field with its default value.

Parameters

- **TIM_EncoderInitStruct:** pointer to a LL_TIM_ENCODER_InitTypeDef structure (encoder interface configuration data structure)

Return values

- **None:**

LL_TIM_ENCODER_Init

Function name

ErrorStatus LL_TIM_ENCODER_Init (TIM_TypeDef * TIMx, LL_TIM_ENCODER_InitTypeDef * TIM_EncoderInitStruct)

Function description

Configure the encoder interface of the timer instance.

Parameters

- **TIMx:** Timer Instance
- **TIM_EncoderInitStruct:** pointer to a LL_TIM_ENCODER_InitTypeDef structure (TIMx encoder interface configuration data structure)

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: TIMx registers are de-initialized
  - ERROR: not applicable

LL_TIM_HALLSENSOR_StructInit

Function name

void LL_TIM_HALLSENSOR_StructInit (LL_TIM_HALLSENSOR_InitTypeDef * TIM_HallSensorInitStruct)

Function description

Set the fields of the TIMx Hall sensor interface configuration data structure to their default values.

Parameters

- **TIM_HallSensorInitStruct:** pointer to a LL_TIM_HALLSENSOR_InitTypeDef structure (HALL sensor interface configuration data structure)

Return values

- **None:**

LL_TIM_HALLSENSOR_Init

Function name

ErrorStatus LL_TIM_HALLSENSOR_Init (TIM_TypeDef * TIMx, LL_TIM_HALLSENSOR_InitTypeDef * TIM_HallSensorInitStruct)

Function description

Configure the Hall sensor interface of the timer instance.

Parameters

- **TIMx:** Timer Instance
- **TIM_HallSensorInitStruct:** pointer to a LL_TIM_HALLSENSOR_InitTypeDef structure (TIMx HALL sensor interface configuration data structure)

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: TIMx registers are de-initialized
  - ERROR: not applicable

Notes

- TIMx CH1, CH2 and CH3 inputs connected through a XOR to the TI1 input channel
- TIMx slave mode controller is configured in reset mode. Selected internal trigger is TI1F_ED.
- Channel 1 is configured as input, IC1 is mapped on TRC.
- Captured value stored in TIMx_CCR1 correspond to the time elapsed between 2 changes on the inputs. It gives information about motor speed.
- Channel 2 is configured in output PWM 2 mode.
- Compare value stored in TIMx_CCR2 corresponds to the commutation delay.
- OC2REF is selected as trigger output on TRGO.
- LL_TIM_IC_POLARITY_BOTHEDGE must not be used for TI1 when it is used when TIMx operates in Hall sensor interface mode.

LL_TIM_BDTR_StructInit

Function name

void LL_TIM_BDTR_StructInit (LL_TIM_BDTR_InitTypeDef * TIM_BDTRInitStruct)

Function description

Set the fields of the Break and Dead Time configuration data structure to their default values.

Parameters

- **TIM_BDTRInitStruct:** pointer to a LL_TIM_BDTR_InitTypeDef structure (Break and Dead Time configuration data structure)

Return values

- **None:**

LL_TIM_BDTR_Init

Function name

ErrorStatus LL_TIM_BDTR_Init (TIM_TypeDef * TIMx, LL_TIM_BDTR_InitTypeDef * TIM_BDTRInitStruct)

Function description

Configure the Break and Dead Time feature of the timer instance.

Parameters

- **TIMx:** Timer Instance
- **TIM_BDTRInitStruct:** pointer to a LL_TIM_BDTR_InitTypeDef structure (Break and Dead Time configuration data structure)

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: Break and Dead Time is initialized
  - ERROR: not applicable

Notes

- As the bits AOE, BKP, BKE, OSSR, OSSI and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.
- Macro IS_TIM_BREAK_INSTANCE(TIMx) can be used to check whether or not a timer instance provides a break input.

92.3 TIM Firmware driver defines

The following section lists the various define and macros of the module.

92.3.1 TIM

TIM

Active Input Selection

LL_TIM_ACTIVEINPUT_DIRECTTI

ICx is mapped on TIx

LL_TIM_ACTIVEINPUT_INDIRECTTI

ICx is mapped on TIy

LL_TIM_ACTIVEINPUT_TRC

ICx is mapped on TRC

Automatic output enable

LL_TIM_AUTOMATICOUTPUT_DISABLE

MOE can be set only by software

LL_TIM_AUTOMATICOUTPUT_ENABLE

MOE can be set by software or automatically at the next update event

Break Enable

LL_TIM_BREAK_DISABLE

Break function disabled

LL_TIM_BREAK_ENABLE

Break function enabled

break polarity

LL_TIM_BREAK_POLARITY_LOW

Break input BRK is active low

LL_TIM_BREAK_POLARITY_HIGH

Break input BRK is active high

Capture Compare DMA Request

LL_TIM_CCDMAREQUEST_CC

CCx DMA request sent when CCx event occurs

LL_TIM_CCDMAREQUEST_UPDATE

CCx DMA requests sent when update event occurs

Capture Compare Update Source

LL_TIM_CCUPDATESOURCE_COMG_ONLY

Capture/compare control bits are updated by setting the COMG bit only

LL_TIM_CCUPDATESOURCE_COMG_AND_TRGI

Capture/compare control bits are updated by setting the COMG bit or when a rising edge occurs on trigger input (TRGI)

Channel

LL_TIM_CHANNEL_CH1

Timer input/output channel 1

LL_TIM_CHANNEL_CH1N

Timer complementary output channel 1

LL_TIM_CHANNEL_CH2

Timer input/output channel 2

LL_TIM_CHANNEL_CH2N

Timer complementary output channel 2

LL_TIM_CHANNEL_CH3

Timer input/output channel 3

LL_TIM_CHANNEL_CH3N

Timer complementary output channel 3

LL_TIM_CHANNEL_CH4

Timer input/output channel 4

Clock Division

LL_TIM_CLOCKDIVISION_DIV1

tDTS=tCK_INT

LL_TIM_CLOCKDIVISION_DIV2

tDTS=2*tCK_INT

LL_TIM_CLOCKDIVISION_DIV4

tDTS=4*tCK_INT

Clock Source

LL_TIM_CLOCKSOURCE_INTERNAL

The timer is clocked by the internal clock provided from the RCC

LL_TIM_CLOCKSOURCE_EXT_MODE1

Counter counts at each rising or falling edge on a selected input

LL_TIM_CLOCKSOURCE_EXT_MODE2

Counter counts at each rising or falling edge on the external trigger input ETR

Counter Direction

LL_TIM_COUNTERDIRECTION_UP

Timer counter counts up

LL_TIM_COUNTERDIRECTION_DOWN

Timer counter counts down

Counter Mode

LL_TIM_COUNTERMODE_UP

Counter used as upcounter

LL_TIM_COUNTERMODE_DOWN

Counter used as downcounter

LL_TIM_COUNTERMODE_CENTER_DOWN

The counter counts up and down alternatively. Output compare interrupt flags of output channels are set only when the counter is counting down.

LL_TIM_COUNTERMODE_CENTER_UP

The counter counts up and down alternatively. Output compare interrupt flags of output channels are set only when the counter is counting up

LL_TIM_COUNTERMODE_CENTER_UP_DOWN

The counter counts up and down alternatively. Output compare interrupt flags of output channels are set only when the counter is counting up or down.

DMA Burst Base Address

LL_TIM_DMABURST_BASEADDR_CR1

TIMx_CR1 register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CR2

TIMx_CR2 register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_SMCR

TIMx_SMCR register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_DIER

TIMx_DIER register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_SR

TIMx_SR register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_EGR

TIMx_EGR register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CCMR1

TIMx_CCMR1 register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CCMR2

TIMx_CCMR2 register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CCER

TIMx_CCER register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CNT

TIMx_CNT register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_PSC

TIMx_PSC register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_ARR

TIMx_ARR register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_RCR

TIMx_RCR register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CCR1

TIMx_CCR1 register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CCR2

TIMx_CCR2 register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CCR3

TIMx_CCR3 register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_CCR4

TIMx_CCR4 register is the DMA base address for DMA burst

LL_TIM_DMABURST_BASEADDR_BDTR

TIMx_BDTR register is the DMA base address for DMA burst

DMA Burst Length

LL_TIM_DMABURST_LENGTH_1TRANSFER

Transfer is done to 1 register starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_2TRANSFERS

Transfer is done to 2 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_3TRANSFERS

Transfer is done to 3 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_4TRANSFERS

Transfer is done to 4 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_5TRANSFERS

Transfer is done to 5 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_6TRANSFERS

Transfer is done to 6 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_7TRANSFERS

Transfer is done to 7 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_8TRANSFERS

Transfer is done to 1 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_9TRANSFERS

Transfer is done to 9 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_10TRANSFERS

Transfer is done to 10 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_11TRANSFERS

Transfer is done to 11 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_12TRANSFERS

Transfer is done to 12 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_13TRANSFERS

Transfer is done to 13 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_14TRANSFERS

Transfer is done to 14 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_15TRANSFERS

Transfer is done to 15 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_16TRANSFERS

Transfer is done to 16 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_17TRANSFERS

Transfer is done to 17 registers starting from the DMA burst base address

LL_TIM_DMABURST_LENGTH_18TRANSFERS

Transfer is done to 18 registers starting from the DMA burst base address

Encoder Mode

LL_TIM_ENCODERMODE_X2_TI1

Quadrature encoder mode 1, x2 mode - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level

LL_TIM_ENCODERMODE_X2_TI2

Quadrature encoder mode 2, x2 mode - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level

LL_TIM_ENCODERMODE_X4_TI12

Quadrature encoder mode 3, x4 mode - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input

External Trigger Filter

LL_TIM_ETR_FILTER_FDIV1

No filter, sampling is done at fDTS

LL_TIM_ETR_FILTER_FDIV1_N2

fSAMPLING=fCK_INT, N=2

LL_TIM_ETR_FILTER_FDIV1_N4

fSAMPLING=fCK_INT, N=4

LL_TIM_ETR_FILTER_FDIV1_N8

fSAMPLING=fCK_INT, N=8

LL_TIM_ETR_FILTER_FDIV2_N6

fSAMPLING=fDTS/2, N=6

LL_TIM_ETR_FILTER_FDIV2_N8

fSAMPLING=fDTS/2, N=8

LL_TIM_ETR_FILTER_FDIV4_N6

fSAMPLING=fDTS/4, N=6

LL_TIM_ETR_FILTER_FDIV4_N8

fSAMPLING=fDTS/4, N=8

LL_TIM_ETR_FILTER_FDIV8_N6

fSAMPLING=fDTS/8, N=8

LL_TIM_ETR_FILTER_FDIV8_N8

fSAMPLING=fDTS/16, N=5

LL_TIM_ETR_FILTER_FDIV16_N5

fSAMPLING=fDTS/16, N=6

LL_TIM_ETR_FILTER_FDIV16_N6

fSAMPLING=fDTS/16, N=8

LL_TIM_ETR_FILTER_FDIV16_N8

fSAMPLING=fDTS/16, N=5

LL_TIM_ETR_FILTER_FDIV32_N5

fSAMPLING=fDTS/32, N=5

LL_TIM_ETR_FILTER_FDIV32_N6

fSAMPLING=fDTS/32, N=6

LL_TIM_ETR_FILTER_FDIV32_N8

fSAMPLING=fDTS/32, N=8

External Trigger Polarity

LL_TIM_ETR_POLARITY_NONINVERTED

ETR is non-inverted, active at high level or rising edge

LL_TIM_ETR_POLARITY_INVERTED

ETR is inverted, active at low level or falling edge

External Trigger Prescaler

LL_TIM_ETR_PRESCALER_DIV1

ETR prescaler OFF

LL_TIM_ETR_PRESCALER_DIV2

ETR frequency is divided by 2

LL_TIM_ETR_PRESCALER_DIV4

ETR frequency is divided by 4

LL_TIM_ETR_PRESCALER_DIV8

ETR frequency is divided by 8

Get Flags Defines

LL_TIM_SR_UIF

Update interrupt flag

LL_TIM_SR_CC1IF

Capture/compare 1 interrupt flag

LL_TIM_SR_CC2IF

Capture/compare 2 interrupt flag

LL_TIM_SR_CC3IF

Capture/compare 3 interrupt flag

LL_TIM_SR_CC4IF

Capture/compare 4 interrupt flag

LL_TIM_SR_COMIF

COM interrupt flag

LL_TIM_SR_TIF

Trigger interrupt flag

LL_TIM_SR_BIF

Break interrupt flag

LL_TIM_SR_CC1OF

Capture/Compare 1 overcapture flag

LL_TIM_SR_CC2OF

Capture/Compare 2 overcapture flag

LL_TIM_SR_CC3OF

Capture/Compare 3 overcapture flag

LL_TIM_SR_CC4OF

Capture/Compare 4 overcapture flag

Input Configuration Prescaler

LL_TIM_ICPSC_DIV1

No prescaler, capture is done each time an edge is detected on the capture input

LL_TIM_ICPSC_DIV2

Capture is done once every 2 events

LL_TIM_ICPSC_DIV4

Capture is done once every 4 events

LL_TIM_ICPSC_DIV8

Capture is done once every 8 events

Input Configuration Filter

LL_TIM_IC_FILTER_FDIV1

No filter, sampling is done at fDTS

LL_TIM_IC_FILTER_FDIV1_N2

fSAMPLING=fCK_INT, N=2

LL_TIM_IC_FILTER_FDIV1_N4

fSAMPLING=fCK_INT, N=4

LL_TIM_IC_FILTER_FDIV1_N8

fSAMPLING=fCK_INT, N=8

LL_TIM_IC_FILTER_FDIV2_N6

fSAMPLING=fDTS/2, N=6

LL_TIM_IC_FILTER_FDIV2_N8

fSAMPLING=fDTS/2, N=8

LL_TIM_IC_FILTER_FDIV4_N6

fSAMPLING=fDTS/4, N=6

LL_TIM_IC_FILTER_FDIV4_N8

fSAMPLING=fDTS/4, N=8

LL_TIM_IC_FILTER_FDIV8_N6

fSAMPLING=fDTS/8, N=6

LL_TIM_IC_FILTER_FDIV8_N8

fSAMPLING=fDTS/8, N=8

LL_TIM_IC_FILTER_FDIV16_N5

fSAMPLING=fDTS/16, N=5

LL_TIM_IC_FILTER_FDIV16_N6

fSAMPLING=fDTS/16, N=6

LL_TIM_IC_FILTER_FDIV16_N8

fSAMPLING=fDTS/16, N=8

LL_TIM_IC_FILTER_FDIV32_N5

fSAMPLING=fDTS/32, N=5

LL_TIM_IC_FILTER_FDIV32_N6

fSAMPLING=fDTS/32, N=6

LL_TIM_IC_FILTER_FDIV32_N8

fSAMPLING=fDTS/32, N=8

Input Configuration Polarity

LL_TIM_IC_POLARITY_RISING

The circuit is sensitive to TIxFP1 rising edge, TIxFP1 is not inverted

LL_TIM_IC_POLARITY_FALLING

The circuit is sensitive to TIxFP1 falling edge, TIxFP1 is inverted

LL_TIM_IC_POLARITY_BOTHEDGE

The circuit is sensitive to both TIxFP1 rising and falling edges, TIxFP1 is not inverted

IT Defines

LL_TIM_DIER_UIE

Update interrupt enable

LL_TIM_DIER_CC1IE

Capture/compare 1 interrupt enable

LL_TIM_DIER_CC2IE

Capture/compare 2 interrupt enable

LL_TIM_DIER_CC3IE

Capture/compare 3 interrupt enable

LL_TIM_DIER_CC4IE

Capture/compare 4 interrupt enable

LL_TIM_DIER_COMIE

COM interrupt enable

LL_TIM_DIER_TIE

Trigger interrupt enable

LL_TIM_DIER_BIE

Break interrupt enable

Lock Level

LL_TIM_LOCKLEVEL_OFF

LOCK OFF - No bit is write protected

LL_TIM_LOCKLEVEL_1

LOCK Level 1

LL_TIM_LOCKLEVEL_2

LOCK Level 2

LL_TIM_LOCKLEVEL_3

LOCK Level 3

Output Configuration Idle State

LL_TIM_OCIDLESTATE_LOW

OCx=0 (after a dead-time if OC is implemented) when MOE=0

LL_TIM_OCIDLESTATE_HIGH

OCx=1 (after a dead-time if OC is implemented) when MOE=0

Output Configuration Mode

LL_TIM_OCMODE_FROZEN

The comparison between the output compare register TIMx_CCRy and the counter TIMx_CNT has no effect on the output channel level

LL_TIM_OCMODE_ACTIVE

OCyREF is forced high on compare match

LL_TIM_OCMODE_INACTIVE

OCyREF is forced low on compare match

LL_TIM_OCMODE_TOGGLE

OCyREF toggles on compare match

LL_TIM_OCMODE_FORCED_INACTIVE

OCyREF is forced low

LL_TIM_OCMODE_FORCED_ACTIVE

OCyREF is forced high

LL_TIM_OCMODE_PWM1

In upcounting, channel y is active as long as TIMx_CNT<TIMx_CCRy else inactive. In downcounting, channel y is inactive as long as TIMx_CNT>TIMx_CCRy else active.

LL_TIM_OCMODE_PWM2

In upcounting, channel y is inactive as long as TIMx_CNT<TIMx_CCRy else active. In downcounting, channel y is active as long as TIMx_CNT>TIMx_CCRy else inactive

Output Configuration Polarity

LL_TIM_OCPOLARITY_HIGH

OCxactive high

LL_TIM_OCPOLARITY_LOW

OCxactive low

Output Configuration State

LL_TIM_OCSTATE_DISABLE

OCx is not active

LL_TIM_OCSTATE_ENABLE

OCx signal is output on the corresponding output pin

One Pulse Mode

LL_TIM_ONEPULSEMODE_SINGLE

Counter is not stopped at update event

LL_TIM_ONEPULSEMODE_REPETITIVE

Counter stops counting at the next update event

OSSI

LL_TIM_OSSI_DISABLE

When inactive, OCx/OCxN outputs are disabled

LL_TIM_OSSI_ENABLE

When inactive, OxC/OCxN outputs are first forced with their inactive level then forced to their idle level after the deadtime

OSSR

LL_TIM_OSSR_DISABLE

When inactive, OCx/OCxN outputs are disabled

LL_TIM_OSSR_ENABLE

When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1

Slave Mode

LL_TIM_SLAVEMODE_DISABLED

Slave mode disabled

LL_TIM_SLAVEMODE_RESET

Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter

LL_TIM_SLAVEMODE_GATED

Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high

LL_TIM_SLAVEMODE_TRIGGER

Trigger Mode - The counter starts at a rising edge of the trigger TRGI

TIM11 External Input Capture 1 Remap

LL_TIM_TIM11_TI1_RMP_GPIO

TIM11 channel 1 is connected to GPIO

LL_TIM_TIM11_TI1_RMP_GPIO1

TIM11 channel 1 is connected to GPIO

LL_TIM_TIM11_TI1_RMP_GPIO2

TIM11 channel 1 is connected to GPIO

LL_TIM_TIM11_TI1_RMP_HSE_RTC

TIM11 channel 1 is connected to HSE_RTC

TIM2 Internal Trigger1 Remap TIM8

LL_TIM_TIM2_ITR1_RMP_TIM8_TRGO

TIM2_ITR1 is connected to TIM8_TRGO

LL_TIM_TIM2_ITR1_RMP_ETH_PTP

TIM2_ITR1 is connected to ETH_PTP

LL_TIM_TIM2_ITR1_RMP_OTG_FS_SOF

TIM2_ITR1 is connected to OTG_FS SOF

LL_TIM_TIM2_ITR1_RMP_OTG_HS_SOF

TIM2_ITR1 is connected to OTG_HS SOF

TIM5 External Input Ch4 Remap

LL_TIM_TIM5_TI4_RMP_GPIO

TIM5 channel 4 is connected to GPIO

LL_TIM_TIM5_TI4_RMP_LSI

TIM5 channel 4 is connected to LSI internal clock

LL_TIM_TIM5_TI4_RMP_LSE

TIM5 channel 4 is connected to LSE

LL_TIM_TIM5_TI4_RMP_RTC

TIM5 channel 4 is connected to RTC wakeup interrupt

Trigger Output

LL_TIM_TRGO_RESET

UG bit from the TIMx_EGR register is used as trigger output

LL_TIM_TRGO_ENABLE

Counter Enable signal (CNT_EN) is used as trigger output

LL_TIM_TRGO_UPDATE

Update event is used as trigger output

LL_TIM_TRGO_CC1IF

CC1 capture or a compare match is used as trigger output

LL_TIM_TRGO_OC1REF

OC1REF signal is used as trigger output

LL_TIM_TRGO_OC2REF

OC2REF signal is used as trigger output

LL_TIM_TRGO_OC3REF

OC3REF signal is used as trigger output

LL_TIM_TRGO_OC4REF

OC4REF signal is used as trigger output

Trigger Selection

LL_TIM_TS_ITR0

Internal Trigger 0 (ITR0) is used as trigger input

LL_TIM_TS_ITR1

Internal Trigger 1 (ITR1) is used as trigger input

LL_TIM_TS_ITR2

Internal Trigger 2 (ITR2) is used as trigger input

LL_TIM_TS_ITR3

Internal Trigger 3 (ITR3) is used as trigger input

LL_TIM_TS_TI1F_ED

TI1 Edge Detector (TI1F_ED) is used as trigger input

LL_TIM_TS_TI1FP1

Filtered Timer Input 1 (TI1FP1) is used as trigger input

LL_TIM_TS_TI2FP2

Filtered Timer Input 2 (TI12P2) is used as trigger input

LL_TIM_TS_ETRF

Filtered external Trigger (ETRF) is used as trigger input

Update Source

LL_TIM_UPDATESOURCE_REGULAR

Counter overflow/underflow, Setting the UG bit or Update generation through the slave mode controller generates an update request

LL_TIM_UPDATESOURCE_COUNTER

Only counter overflow/underflow generates an update request

Exported_Macros

__LL_TIM_CALC_DEADTIME

Description:

- HELPER macro calculating DTG[0:7] in the TIMx_BDTR register to achieve the requested dead time duration.

Parameters:

- __TIMCLK__: timer input clock frequency (in Hz)
- __CKD__: This parameter can be one of the following values:
  - LL_TIM_CLOCKDIVISION_DIV1
  - LL_TIM_CLOCKDIVISION_DIV2
  - LL_TIM_CLOCKDIVISION_DIV4
- __DT__: deadtime duration (in ns)

Return value:

- DTG[0:7]

Notes:

- ex: __LL_TIM_CALC_DEADTIME (80000000, LL_TIM_GetClockDivision (), 120);

__LL_TIM_CALC_PSC

Description:

- HELPER macro calculating the prescaler value to achieve the required counter clock frequency.

Parameters:

- __TIMCLK__: timer input clock frequency (in Hz)
- __CNTCLK__: counter clock frequency (in Hz)

Return value:

- Prescaler: value (between Min_Data=0 and Max_Data=65535)

Notes:

- ex: __LL_TIM_CALC_PSC (80000000, 1000000);

__LL_TIM_CALC_ARR

Description:

- HELPER macro calculating the auto-reload value to achieve the required output signal frequency.

Parameters:

- __TIMCLK__: timer input clock frequency (in Hz)
- __PSC__: prescaler
- __FREQ__: output signal frequency (in Hz)

Return value:

- Auto-reload: value (between Min_Data=0 and Max_Data=65535)

Notes:

- ex: __LL_TIM_CALC_ARR (1000000, LL_TIM_GetPrescaler (), 10000);

__LL_TIM_CALC_DELAY

Description:

- HELPER macro calculating the compare value required to achieve the required timer output compare active/inactive delay.

Parameters:

- __TIMCLK__: timer input clock frequency (in Hz)
- __PSC__: prescaler
- __DELAY__: timer output compare active/inactive delay (in us)

Return value:

- Compare: value (between Min_Data=0 and Max_Data=65535)

Notes:

- ex: __LL_TIM_CALC_DELAY (1000000, LL_TIM_GetPrescaler (), 10);

__LL_TIM_CALC_PULSE

Description:

- HELPER macro calculating the auto-reload value to achieve the required pulse duration (when the timer operates in one pulse mode).

Parameters:

- __TIMCLK__: timer input clock frequency (in Hz)
- __PSC__: prescaler
- __DELAY__: timer output compare active/inactive delay (in us)
- __PULSE__: pulse duration (in us)

Return value:

- Auto-reload: value (between Min_Data=0 and Max_Data=65535)

Notes:

- ex: __LL_TIM_CALC_PULSE (1000000, LL_TIM_GetPrescaler (), 10, 20);

__LL_TIM_GET_ICPSC_RATIO

Description:

- HELPER macro retrieving the ratio of the input capture prescaler.

Parameters:

- __ICPSC__: This parameter can be one of the following values:
  - LL_TIM_ICPSC_DIV1
  - LL_TIM_ICPSC_DIV2
  - LL_TIM_ICPSC_DIV4
  - LL_TIM_ICPSC_DIV8

Return value:

- Input: capture prescaler ratio (1, 2, 4 or 8)

Notes:

- ex: __LL_TIM_GET_ICPSC_RATIO (LL_TIM_IC_GetPrescaler ());

Common Write and read registers Macros

LL_TIM_WriteReg

Description:

- Write a value in TIM register.

Parameters:

- __INSTANCE__: TIM Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_TIM_ReadReg

Description:

- Read a value in TIM register.

Parameters:

- __INSTANCE__: TIM Instance
- __REG__: Register to be read

Return value:

- Register: value

93 LL USART Generic Driver

93.1 USART Firmware driver registers structures

93.1.1 LL_USART_InitTypeDef

LL_USART_InitTypeDef is defined in the stm32f4xx_ll_usart.h

Data Fields

- ***uint32_t BaudRate***
- ***uint32_t DataWidth***
- ***uint32_t StopBits***
- ***uint32_t Parity***
- ***uint32_t TransferDirection***
- ***uint32_t HardwareFlowControl***
- ***uint32_t OverSampling***

Field Documentation

- ***uint32_t LL_USART_InitTypeDef::BaudRate***

  This field defines expected Usart communication baud rate.This feature can be modified afterwards using unitary function **LL_USART_SetBaudRate()**.
- ***uint32_t LL_USART_InitTypeDef::DataWidth***

  Specifies the number of data bits transmitted or received in a frame. This parameter can be a value of ***USART_LL_EC_DATAWIDTH***.This feature can be modified afterwards using unitary function **LL_USART_SetDataWidth()**.
- ***uint32_t LL_USART_InitTypeDef::StopBits***

  Specifies the number of stop bits transmitted. This parameter can be a value of ***USART_LL_EC_STOPBITS***.This feature can be modified afterwards using unitary function **LL_USART_SetStopBitsLength()**.
- ***uint32_t LL_USART_InitTypeDef::Parity***

  Specifies the parity mode. This parameter can be a value of ***USART_LL_EC_PARITY***.This feature can be modified afterwards using unitary function **LL_USART_SetParity()**.
- ***uint32_t LL_USART_InitTypeDef::TransferDirection***

  Specifies whether the Receive and/or Transmit mode is enabled or disabled. This parameter can be a value of ***USART_LL_EC_DIRECTION***.This feature can be modified afterwards using unitary function **LL_USART_SetTransferDirection()**.
- ***uint32_t LL_USART_InitTypeDef::HardwareFlowControl***

  Specifies whether the hardware flow control mode is enabled or disabled. This parameter can be a value of ***USART_LL_EC_HWCONTROL***.This feature can be modified afterwards using unitary function **LL_USART_SetHWFlowCtrl()**.
- ***uint32_t LL_USART_InitTypeDef::OverSampling***

  Specifies whether USART oversampling mode is 16 or 8. This parameter can be a value of ***USART_LL_EC_OVERSAMPLING***.This feature can be modified afterwards using unitary function **LL_USART_SetOverSampling()**.

93.1.2 LL_USART_ClockInitTypeDef

LL_USART_ClockInitTypeDef is defined in the stm32f4xx_ll_usart.h

Data Fields

- ***uint32_t ClockOutput***
- ***uint32_t ClockPolarity***
- ***uint32_t ClockPhase***
- ***uint32_t LastBitClockPulse***

Field Documentation

- ***uint32_t LL_USART_ClockInitTypeDef::ClockOutput***
  Specifies whether the USART clock is enabled or disabled. This parameter can be a value of ***USART_LL_EC_CLOCK***.USART HW configuration can be modified afterwards using unitary functions **LL_USART_EnableSCLKOutput()** or **LL_USART_DisableSCLKOutput()**. For more details, refer to description of this function.
- ***uint32_t LL_USART_ClockInitTypeDef::ClockPolarity***
  Specifies the steady state of the serial clock. This parameter can be a value of ***USART_LL_EC_POLARITY***.USART HW configuration can be modified afterwards using unitary functions **LL_USART_SetClockPolarity()**. For more details, refer to description of this function.
- ***uint32_t LL_USART_ClockInitTypeDef::ClockPhase***
  Specifies the clock transition on which the bit capture is made. This parameter can be a value of ***USART_LL_EC_PHASE***.USART HW configuration can be modified afterwards using unitary functions **LL_USART_SetClockPhase()**. For more details, refer to description of this function.
- ***uint32_t LL_USART_ClockInitTypeDef::LastBitClockPulse***
  Specifies whether the clock pulse corresponding to the last transmitted data bit (MSB) has to be output on the SCLK pin in synchronous mode. This parameter can be a value of ***USART_LL_EC_LASTCLKPULSE***.USART HW configuration can be modified afterwards using unitary functions **LL_USART_SetLastClkPulseOutput()**. For more details, refer to description of this function.

93.2 USART Firmware driver API description

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

93.2.1 Detailed description of functions

LL_USART_Enable

Function name

__STATIC_INLINE void LL_USART_Enable (USART_TypeDef * USARTx)

Function description

USART Enable.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 UE LL_USART_Enable

LL_USART_Disable

Function name

__STATIC_INLINE void LL_USART_Disable (USART_TypeDef * USARTx)

Function description

USART Disable (all USART prescalers and outputs are disabled)

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- When USART is disabled, USART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the USART is kept, but all the status flags, in the USARTx_SR are set to their default values.

Reference Manual to LL API cross reference:

- CR1 UE LL_USART_Disable

LL_USART_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabled (USART_TypeDef * USARTx)

Function description

Indicate if USART is enabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 UE LL_USART_IsEnabled

LL_USART_EnableDirectionRx

Function name

__STATIC_INLINE void LL_USART_EnableDirectionRx (USART_TypeDef * USARTx)

Function description

Receiver Enable (Receiver is enabled and begins searching for a start bit)

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RE LL_USART_EnableDirectionRx

LL_USART_DisableDirectionRx

Function name

__STATIC_INLINE void LL_USART_DisableDirectionRx (USART_TypeDef * USARTx)

Function description

Receiver Disable.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RE LL_USART_DisableDirectionRx

LL_USART_EnableDirectionTx

Function name

__STATIC_INLINE void LL_USART_EnableDirectionTx (USART_TypeDef * USARTx)

Function description

Transmitter Enable.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TE LL_USART_EnableDirectionTx

LL_USART_DisableDirectionTx

Function name

__STATIC_INLINE void LL_USART_DisableDirectionTx (USART_TypeDef * USARTx)

Function description

Transmitter Disable.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TE LL_USART_DisableDirectionTx

LL_USART_SetTransferDirection

Function name

__STATIC_INLINE void LL_USART_SetTransferDirection (USART_TypeDef * USARTx, uint32_t TransferDirection)

Function description

Configure simultaneously enabled/disabled states of Transmitter and Receiver.

Parameters

- **USARTx:** USART Instance
- **TransferDirection:** This parameter can be one of the following values:
  - LL_USART_DIRECTION_NONE
  - LL_USART_DIRECTION_RX
  - LL_USART_DIRECTION_TX
  - LL_USART_DIRECTION_TX_RX

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RE LL_USART_SetTransferDirection
- CR1 TE LL_USART_SetTransferDirection

LL_USART_GetTransferDirection

Function name

__STATIC_INLINE uint32_t LL_USART_GetTransferDirection (USART_TypeDef * USARTx)

Function description

Return enabled/disabled states of Transmitter and Receiver.

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_DIRECTION_NONE
  - LL_USART_DIRECTION_RX
  - LL_USART_DIRECTION_TX
  - LL_USART_DIRECTION_TX_RX

Reference Manual to LL API cross reference:

- CR1 RE LL_USART_GetTransferDirection
- CR1 TE LL_USART_GetTransferDirection

LL_USART_SetParity

Function name

__STATIC_INLINE void LL_USART_SetParity (USART_TypeDef * USARTx, uint32_t Parity)

Function description

Configure Parity (enabled/disabled and parity mode if enabled).

Parameters

- **USARTx:** USART Instance
- **Parity:** This parameter can be one of the following values:
  - LL_USART_PARITY_NONE
  - LL_USART_PARITY_EVEN
  - LL_USART_PARITY_ODD

Return values

- **None:**

Notes

- This function selects if hardware parity control (generation and detection) is enabled or disabled. When the parity control is enabled (Odd or Even), computed parity bit is inserted at the MSB position (9th or 8th bit depending on data width) and parity is checked on the received data.

Reference Manual to LL API cross reference:

- CR1 PS LL_USART_SetParity
- CR1 PCE LL_USART_SetParity

LL_USART_GetParity

Function name

__STATIC_INLINE uint32_t LL_USART_GetParity (USART_TypeDef * USARTx)

Function description

Return Parity configuration (enabled/disabled and parity mode if enabled)

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_PARITY_NONE
  - LL_USART_PARITY_EVEN
  - LL_USART_PARITY_ODD

Reference Manual to LL API cross reference:

- CR1 PS LL_USART_GetParity
- CR1 PCE LL_USART_GetParity

LL_USART_SetWakeUpMethod

Function name

__STATIC_INLINE void LL_USART_SetWakeUpMethod (USART_TypeDef * USARTx, uint32_t Method)

Function description

Set Receiver Wake Up method from Mute mode.

Parameters

- **USARTx:** USART Instance
- **Method:** This parameter can be one of the following values:
  - LL_USART_WAKEUP_IDLELINE
  - LL_USART_WAKEUP_ADDRESSMARK

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 WAKE LL_USART_SetWakeUpMethod

LL_USART_GetWakeUpMethod

Function name

__STATIC_INLINE uint32_t LL_USART_GetWakeUpMethod (USART_TypeDef * USARTx)

Function description

Return Receiver Wake Up method from Mute mode.

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_WAKEUP_IDLELINE
  - LL_USART_WAKEUP_ADDRESSMARK

Reference Manual to LL API cross reference:

- CR1 WAKE LL_USART_GetWakeUpMethod

LL_USART_SetDataWidth

Function name

__STATIC_INLINE void LL_USART_SetDataWidth (USART_TypeDef * USARTx, uint32_t DataWidth)

Function description

Set Word length (i.e.

Parameters

- **USARTx:** USART Instance
- **DataWidth:** This parameter can be one of the following values:
  - LL_USART_DATAWIDTH_8B
  - LL_USART_DATAWIDTH_9B

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 M LL_USART_SetDataWidth

LL_USART_GetDataWidth

Function name

__STATIC_INLINE uint32_t LL_USART_GetDataWidth (USART_TypeDef * USARTx)

Function description

Return Word length (i.e.

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_DATAWIDTH_8B
  - LL_USART_DATAWIDTH_9B

Reference Manual to LL API cross reference:

- CR1 M LL_USART_GetDataWidth

LL_USART_SetOverSampling

Function name

__STATIC_INLINE void LL_USART_SetOverSampling (USART_TypeDef * USARTx, uint32_t OverSampling)

Function description

Set Oversampling to 8-bit or 16-bit mode.

Parameters

- **USARTx:** USART Instance
- **OverSampling:** This parameter can be one of the following values:
  - LL_USART_OVERSAMPLING_16
  - LL_USART_OVERSAMPLING_8

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 OVER8 LL_USART_SetOverSampling

LL_USART_GetOverSampling

Function name

__STATIC_INLINE uint32_t LL_USART_GetOverSampling (USART_TypeDef * USARTx)

Function description

Return Oversampling mode.

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_OVERSAMPLING_16
  - LL_USART_OVERSAMPLING_8

Reference Manual to LL API cross reference:

- CR1 OVER8 LL_USART_GetOverSampling

LL_USART_SetLastClkPulseOutput

Function name

__STATIC_INLINE void LL_USART_SetLastClkPulseOutput (USART_TypeDef * USARTx, uint32_t LastBitClockPulse)

Function description

Configure if Clock pulse of the last data bit is output to the SCLK pin or not.

Parameters

- **USARTx:** USART Instance
- **LastBitClockPulse:** This parameter can be one of the following values:
  - LL_USART_LASTCLKPULSE_NO_OUTPUT
  - LL_USART_LASTCLKPULSE_OUTPUT

Return values

- **None:**

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LBCL LL_USART_SetLastClkPulseOutput

LL_USART_GetLastClkPulseOutput

Function name

__STATIC_INLINE uint32_t LL_USART_GetLastClkPulseOutput (USART_TypeDef * USARTx)

Function description

Retrieve Clock pulse of the last data bit output configuration (Last bit Clock pulse output to the SCLK pin or not)

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_LASTCLKPULSE_NO_OUTPUT
  - LL_USART_LASTCLKPULSE_OUTPUT

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LBCL LL_USART_GetLastClkPulseOutput

LL_USART_SetClockPhase

Function name

__STATIC_INLINE void LL_USART_SetClockPhase (USART_TypeDef * USARTx, uint32_t ClockPhase)

Function description

Select the phase of the clock output on the SCLK pin in synchronous mode.

Parameters

- **USARTx:** USART Instance
- **ClockPhase:** This parameter can be one of the following values:
  - LL_USART_PHASE_1EDGE
  - LL_USART_PHASE_2EDGE

Return values

- **None:**

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 CPHA LL_USART_SetClockPhase

LL_USART_GetClockPhase

Function name

__STATIC_INLINE uint32_t LL_USART_GetClockPhase (USART_TypeDef * USARTx)

Function description

Return phase of the clock output on the SCLK pin in synchronous mode.

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_PHASE_1EDGE
  - LL_USART_PHASE_2EDGE

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 CPHA LL_USART_GetClockPhase

LL_USART_SetClockPolarity

Function name

__STATIC_INLINE void LL_USART_SetClockPolarity (USART_TypeDef * USARTx, uint32_t ClockPolarity)

Function description

Select the polarity of the clock output on the SCLK pin in synchronous mode.

Parameters

- **USARTx:** USART Instance
- **ClockPolarity:** This parameter can be one of the following values:
  - LL_USART_POLARITY_LOW
  - LL_USART_POLARITY_HIGH

Return values

- **None:**

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 CPOL LL_USART_SetClockPolarity

LL_USART_GetClockPolarity

Function name

__STATIC_INLINE uint32_t LL_USART_GetClockPolarity (USART_TypeDef * USARTx)

Function description

Return polarity of the clock output on the SCLK pin in synchronous mode.

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_POLARITY_LOW
  - LL_USART_POLARITY_HIGH

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 CPOL LL_USART_GetClockPolarity

LL_USART_ConfigClock

Function name

__STATIC_INLINE void LL_USART_ConfigClock (USART_TypeDef * USARTx, uint32_t Phase, uint32_t Polarity, uint32_t LBCPOutput)

Function description

Configure Clock signal format (Phase Polarity and choice about output of last bit clock pulse)

Parameters

- **USARTx:** USART Instance
- **Phase:** This parameter can be one of the following values:
  - LL_USART_PHASE_1EDGE
  - LL_USART_PHASE_2EDGE
- **Polarity:** This parameter can be one of the following values:
  - LL_USART_POLARITY_LOW
  - LL_USART_POLARITY_HIGH
- **LBCPOutput:** This parameter can be one of the following values:
  - LL_USART_LASTCLKPULSE_NO_OUTPUT
  - LL_USART_LASTCLKPULSE_OUTPUT

Return values

- **None:**

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.
- Call of this function is equivalent to following function call sequence : Clock Phase configuration using LL_USART_SetClockPhase() functionClock Polarity configuration using LL_USART_SetClockPolarity() functionOutput of Last bit Clock pulse configuration using LL_USART_SetLastClkPulseOutput() function

Reference Manual to LL API cross reference:

- CR2 CPHA LL_USART_ConfigClock
- CR2 CPOL LL_USART_ConfigClock
- CR2 LBCL LL_USART_ConfigClock

LL_USART_EnableSCLKOutput

Function name

__STATIC_INLINE void LL_USART_EnableSCLKOutput (USART_TypeDef * USARTx)

Function description

Enable Clock output on SCLK pin.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 CLKEN LL_USART_EnableSCLKOutput

LL_USART_DisableSCLKOutput

Function name

__STATIC_INLINE void LL_USART_DisableSCLKOutput (USART_TypeDef * USARTx)

Function description

Disable Clock output on SCLK pin.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 CLKEN LL_USART_DisableSCLKOutput

LL_USART_IsEnabledSCLKOutput

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledSCLKOutput (USART_TypeDef * USARTx)

Function description

Indicate if Clock output on SCLK pin is enabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 CLKEN LL_USART_IsEnabledSCLKOutput

LL_USART_SetStopBitsLength

Function name

__STATIC_INLINE void LL_USART_SetStopBitsLength (USART_TypeDef * USARTx, uint32_t StopBits)

Function description

Set the length of the stop bits.

Parameters

- **USARTx:** USART Instance
- **StopBits:** This parameter can be one of the following values:
  - LL_USART_STOPBITS_0_5
  - LL_USART_STOPBITS_1
  - LL_USART_STOPBITS_1_5
  - LL_USART_STOPBITS_2

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR2 STOP LL_USART_SetStopBitsLength

LL_USART_GetStopBitsLength

Function name

__STATIC_INLINE uint32_t LL_USART_GetStopBitsLength (USART_TypeDef * USARTx)

Function description

Retrieve the length of the stop bits.

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_STOPBITS_0_5
  - LL_USART_STOPBITS_1
  - LL_USART_STOPBITS_1_5
  - LL_USART_STOPBITS_2

Reference Manual to LL API cross reference:

- CR2 STOP LL_USART_GetStopBitsLength

LL_USART_ConfigCharacter

Function name

__STATIC_INLINE void LL_USART_ConfigCharacter (USART_TypeDef * USARTx, uint32_t DataWidth, uint32_t Parity, uint32_t StopBits)

Function description

Configure Character frame format (Datawidth, Parity control, Stop Bits)

Parameters

- **USARTx:** USART Instance
- **DataWidth:** This parameter can be one of the following values:
  - LL_USART_DATAWIDTH_8B
  - LL_USART_DATAWIDTH_9B
- **Parity:** This parameter can be one of the following values:
  - LL_USART_PARITY_NONE
  - LL_USART_PARITY_EVEN
  - LL_USART_PARITY_ODD
- **StopBits:** This parameter can be one of the following values:
  - LL_USART_STOPBITS_0_5
  - LL_USART_STOPBITS_1
  - LL_USART_STOPBITS_1_5
  - LL_USART_STOPBITS_2

Return values

- **None:**

Notes

- Call of this function is equivalent to following function call sequence : Data Width configuration using LL_USART_SetDataWidth() functionParity Control and mode configuration using LL_USART_SetParity() functionStop bits configuration using LL_USART_SetStopBitsLength() function

Reference Manual to LL API cross reference:

- CR1 PS LL_USART_ConfigCharacter
- CR1 PCE LL_USART_ConfigCharacter
- CR1 M LL_USART_ConfigCharacter
- CR2 STOP LL_USART_ConfigCharacter

LL_USART_SetNodeAddress

Function name

__STATIC_INLINE void LL_USART_SetNodeAddress (USART_TypeDef * USARTx, uint32_t NodeAddress)

Function description

Set Address of the USART node.

Parameters

- **USARTx:** USART Instance
- **NodeAddress:** 4 bit Address of the USART node.

Return values

- **None:**

Notes

- This is used in multiprocessor communication during Mute mode or Stop mode, for wake up with address mark detection.

Reference Manual to LL API cross reference:

- CR2 ADD LL_USART_SetNodeAddress

LL_USART_GetNodeAddress

Function name

__STATIC_INLINE uint32_t LL_USART_GetNodeAddress (USART_TypeDef * USARTx)

Function description

Return 4 bit Address of the USART node as set in ADD field of CR2.

Parameters

- **USARTx:** USART Instance

Return values

- **Address:** of the USART node (Value between Min_Data=0 and Max_Data=255)

Notes

- only 4bits (b3-b0) of returned value are relevant (b31-b4 are not relevant)

Reference Manual to LL API cross reference:

- CR2 ADD LL_USART_GetNodeAddress

LL_USART_EnableRTSHWFlowCtrl

Function name

__STATIC_INLINE void LL_USART_EnableRTSHWFlowCtrl (USART_TypeDef * USARTx)

Function description

Enable RTS HW Flow Control.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 RTSE LL_USART_EnableRTSHWFlowCtrl

LL_USART_DisableRTSHWFlowCtrl

Function name

__STATIC_INLINE void LL_USART_DisableRTSHWFlowCtrl (USART_TypeDef * USARTx)

Function description

Disable RTS HW Flow Control.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 RTSE LL_USART_DisableRTSHWFlowCtrl

LL_USART_EnableCTSHWFlowCtrl

Function name

__STATIC_INLINE void LL_USART_EnableCTSHWFlowCtrl (USART_TypeDef * USARTx)

Function description

Enable CTS HW Flow Control.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 CTSE LL_USART_EnableCTSHWFlowCtrl

LL_USART_DisableCTSHWFlowCtrl

Function name

__STATIC_INLINE void LL_USART_DisableCTSHWFlowCtrl (USART_TypeDef * USARTx)

Function description

Disable CTS HW Flow Control.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 CTSE LL_USART_DisableCTSHWFlowCtrl

LL_USART_SetHWFlowCtrl

Function name

__STATIC_INLINE void LL_USART_SetHWFlowCtrl (USART_TypeDef * USARTx, uint32_t HardwareFlowControl)

Function description

Configure HW Flow Control mode (both CTS and RTS)

Parameters

- **USARTx:** USART Instance
- **HardwareFlowControl:** This parameter can be one of the following values:
  - LL_USART_HWCONTROL_NONE
  - LL_USART_HWCONTROL_RTS
  - LL_USART_HWCONTROL_CTS
  - LL_USART_HWCONTROL_RTS_CTS

Return values

- **None:**

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 RTSE LL_USART_SetHWFlowCtrl
- CR3 CTSE LL_USART_SetHWFlowCtrl

LL_USART_GetHWFlowCtrl

Function name

__STATIC_INLINE uint32_t LL_USART_GetHWFlowCtrl (USART_TypeDef * USARTx)

Function description

Return HW Flow Control configuration (both CTS and RTS)

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_HWCONTROL_NONE
  - LL_USART_HWCONTROL_RTS
  - LL_USART_HWCONTROL_CTS
  - LL_USART_HWCONTROL_RTS_CTS

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 RTSE LL_USART_GetHWFlowCtrl
- CR3 CTSE LL_USART_GetHWFlowCtrl

LL_USART_EnableOneBitSamp

Function name

__STATIC_INLINE void LL_USART_EnableOneBitSamp (USART_TypeDef * USARTx)

Function description

Enable One bit sampling method.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR3 ONEBIT LL_USART_EnableOneBitSamp

LL_USART_DisableOneBitSamp

Function name

__STATIC_INLINE void LL_USART_DisableOneBitSamp (USART_TypeDef * USARTx)

Function description

Disable One bit sampling method.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR3 ONEBIT LL_USART_DisableOneBitSamp

LL_USART_IsEnabledOneBitSamp

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledOneBitSamp (USART_TypeDef * USARTx)

Function description

Indicate if One bit sampling method is enabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR3 ONEBIT LL_USART_IsEnabledOneBitSamp

LL_USART_SetBaudRate

Function name

__STATIC_INLINE void LL_USART_SetBaudRate (USART_TypeDef * USARTx, uint32_t PeriphClk, uint32_t OverSampling, uint32_t BaudRate)

Function description

Configure USART BRR register for achieving expected Baud Rate value.

Parameters

- **USARTx:** USART Instance
- **PeriphClk:** Peripheral Clock
- **OverSampling:** This parameter can be one of the following values:
  - LL_USART_OVERSAMPLING_16
  - LL_USART_OVERSAMPLING_8
- **BaudRate:** Baud Rate

Return values

- **None:**

Notes

- Compute and set USARTDIV value in BRR Register (full BRR content) according to used Peripheral Clock, Oversampling mode, and expected Baud Rate values
- Peripheral clock and Baud rate values provided as function parameters should be valid (Baud rate value != 0)

Reference Manual to LL API cross reference:

- BRR BRR LL_USART_SetBaudRate

LL_USART_GetBaudRate

Function name

__STATIC_INLINE uint32_t LL_USART_GetBaudRate (USART_TypeDef * USARTx, uint32_t PeriphClk, uint32_t OverSampling)

Function description

Return current Baud Rate value, according to USARTDIV present in BRR register (full BRR content), and to used Peripheral Clock and Oversampling mode values.

Parameters

- **USARTx:** USART Instance
- **PeriphClk:** Peripheral Clock
- **OverSampling:** This parameter can be one of the following values:
  - LL_USART_OVERSAMPLING_16
  - LL_USART_OVERSAMPLING_8

Return values

- **Baud:** Rate

Notes

- In case of non-initialized or invalid value stored in BRR register, value 0 will be returned.

Reference Manual to LL API cross reference:

- BRR BRR LL_USART_GetBaudRate

LL_USART_EnableIrda

Function name

__STATIC_INLINE void LL_USART_EnableIrda (USART_TypeDef * USARTx)

Function description

Enable IrDA mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 IREN LL_USART_EnableIrda

LL_USART_DisableIrda

Function name

__STATIC_INLINE void LL_USART_DisableIrda (USART_TypeDef * USARTx)

Function description

Disable IrDA mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 IREN LL_USART_DisableIrda

LL_USART_IsEnabledIrda

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIrda (USART_TypeDef * USARTx)

Function description

Indicate if IrDA mode is enabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 IREN LL_USART_IsEnabledIrda

LL_USART_SetIrdaPowerMode

Function name

__STATIC_INLINE void LL_USART_SetIrdaPowerMode (USART_TypeDef * USARTx, uint32_t PowerMode)

Function description

Configure IrDA Power Mode (Normal or Low Power)

Parameters

- **USARTx:** USART Instance
- **PowerMode:** This parameter can be one of the following values:
  - LL_USART_IRDA_POWER_NORMAL
  - LL_USART_IRDA_POWER_LOW

Return values

- **None:**

Notes

- Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 IRLP LL_USART_SetIrdaPowerMode

LL_USART_GetIrdaPowerMode

Function name

__STATIC_INLINE uint32_t LL_USART_GetIrdaPowerMode (USART_TypeDef * USARTx)

Function description

Retrieve IrDA Power Mode configuration (Normal or Low Power)

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_IRDA_POWER_NORMAL
  - LL_USART_PHASE_2EDGE

Notes

- Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 IRLP LL_USART_GetIrdaPowerMode

LL_USART_SetIrdaPrescaler

Function name

__STATIC_INLINE void LL_USART_SetIrdaPrescaler (USART_TypeDef * USARTx, uint32_t PrescalerValue)

Function description

Set Irda prescaler value, used for dividing the USART clock source to achieve the Irda Low Power frequency (8 bits value)

Parameters

- **USARTx:** USART Instance
- **PrescalerValue:** Value between Min_Data=0x00 and Max_Data=0xFF

Return values

- **None:**

Notes

- Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- GTPR PSC LL_USART_SetIrdaPrescaler

LL_USART_GetIrdaPrescaler

Function name

__STATIC_INLINE uint32_t LL_USART_GetIrdaPrescaler (USART_TypeDef * USARTx)

Function description

Return Irda prescaler value, used for dividing the USART clock source to achieve the Irda Low Power frequency (8 bits value)

Parameters

- **USARTx:** USART Instance

Return values

- **Irda:** prescaler value (Value between Min_Data=0x00 and Max_Data=0xFF)

Notes

- Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- GTPR PSC LL_USART_GetIrdaPrescaler

LL_USART_EnableSmartcardNACK

Function name

__STATIC_INLINE void LL_USART_EnableSmartcardNACK (USART_TypeDef * USARTx)

Function description

Enable Smartcard NACK transmission.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 NACK LL_USART_EnableSmartcardNACK

LL_USART_DisableSmartcardNACK

Function name

__STATIC_INLINE void LL_USART_DisableSmartcardNACK (USART_TypeDef * USARTx)

Function description

Disable Smartcard NACK transmission.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 NACK LL_USART_DisableSmartcardNACK

LL_USART_IsEnabledSmartcardNACK

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledSmartcardNACK (USART_TypeDef * USARTx)

Function description

Indicate if Smartcard NACK transmission is enabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 NACK LL_USART_IsEnabledSmartcardNACK

LL_USART_EnableSmartcard

Function name

__STATIC_INLINE void LL_USART_EnableSmartcard (USART_TypeDef * USARTx)

Function description

Enable Smartcard mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 SCEN LL_USART_EnableSmartcard

LL_USART_DisableSmartcard

Function name

__STATIC_INLINE void LL_USART_DisableSmartcard (USART_TypeDef * USARTx)

Function description

Disable Smartcard mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 SCEN LL_USART_DisableSmartcard

LL_USART_IsEnabledSmartcard

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledSmartcard (USART_TypeDef * USARTx)

Function description

Indicate if Smartcard mode is enabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 SCEN LL_USART_IsEnabledSmartcard

LL_USART_SetSmartcardPrescaler

Function name

__STATIC_INLINE void LL_USART_SetSmartcardPrescaler (USART_TypeDef * USARTx, uint32_t PrescalerValue)

Function description

Set Smartcard prescaler value, used for dividing the USART clock source to provide the SMARTCARD Clock (5 bits value)

Parameters

- **USARTx:** USART Instance
- **PrescalerValue:** Value between Min_Data=0 and Max_Data=31

Return values

- **None:**

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- GTPR PSC LL_USART_SetSmartcardPrescaler

LL_USART_GetSmartcardPrescaler

Function name

__STATIC_INLINE uint32_t LL_USART_GetSmartcardPrescaler (USART_TypeDef * USARTx)

Function description

Return Smartcard prescaler value, used for dividing the USART clock source to provide the SMARTCARD Clock (5 bits value)

Parameters

- **USARTx:** USART Instance

Return values

- **Smartcard:** prescaler value (Value between Min_Data=0 and Max_Data=31)

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- GTPR PSC LL_USART_GetSmartcardPrescaler

LL_USART_SetSmartcardGuardTime

Function name

__STATIC_INLINE void LL_USART_SetSmartcardGuardTime (USART_TypeDef * USARTx, uint32_t GuardTime)

Function description

Set Smartcard Guard time value, expressed in nb of baud clocks periods (GT[7:0] bits : Guard time value)

Parameters

- **USARTx:** USART Instance
- **GuardTime:** Value between Min_Data=0x00 and Max_Data=0xFF

Return values

- **None:**

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- GTPR GT LL_USART_SetSmartcardGuardTime

LL_USART_GetSmartcardGuardTime

Function name

__STATIC_INLINE uint32_t LL_USART_GetSmartcardGuardTime (USART_TypeDef * USARTx)

Function description

Return Smartcard Guard time value, expressed in nb of baud clocks periods (GT[7:0] bits : Guard time value)

Parameters

- **USARTx:** USART Instance

Return values

- **Smartcard:** Guard time value (Value between Min_Data=0x00 and Max_Data=0xFF)

Notes

- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- GTPR GT LL_USART_GetSmartcardGuardTime

LL_USART_EnableHalfDuplex

Function name

__STATIC_INLINE void LL_USART_EnableHalfDuplex (USART_TypeDef * USARTx)

Function description

Enable Single Wire Half-Duplex mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HALFDUPLEX_INSTANCE(USARTx) can be used to check whether or not Half-Duplex mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 HDSEL LL_USART_EnableHalfDuplex

LL_USART_DisableHalfDuplex

Function name

__STATIC_INLINE void LL_USART_DisableHalfDuplex (USART_TypeDef * USARTx)

Function description

Disable Single Wire Half-Duplex mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HALFDUPLEX_INSTANCE(USARTx) can be used to check whether or not Half-Duplex mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 HDSEL LL_USART_DisableHalfDuplex

LL_USART_IsEnabledHalfDuplex

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledHalfDuplex (USART_TypeDef * USARTx)

Function description

Indicate if Single Wire Half-Duplex mode is enabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_UART_HALFDUPLEX_INSTANCE(USARTx) can be used to check whether or not Half-Duplex mode is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 HDSEL LL_USART_IsEnabledHalfDuplex

LL_USART_SetLINBrkDetectionLen

Function name

__STATIC_INLINE void LL_USART_SetLINBrkDetectionLen (USART_TypeDef * USARTx, uint32_t LINBDLength)

Function description

Set LIN Break Detection Length.

Parameters

- **USARTx:** USART Instance
- **LINBDLength:** This parameter can be one of the following values:
  - LL_USART_LINBREAK_DETECT_10B
  - LL_USART_LINBREAK_DETECT_11B

Return values

- **None:**

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LBDL LL_USART_SetLINBrkDetectionLen

LL_USART_GetLINBrkDetectionLen

Function name

__STATIC_INLINE uint32_t LL_USART_GetLINBrkDetectionLen (USART_TypeDef * USARTx)

Function description

Return LIN Break Detection Length.

Parameters

- **USARTx:** USART Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_USART_LINBREAK_DETECT_10B
  - LL_USART_LINBREAK_DETECT_11B

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LBDL LL_USART_GetLINBrkDetectionLen

LL_USART_EnableLIN

Function name

__STATIC_INLINE void LL_USART_EnableLIN (USART_TypeDef * USARTx)

Function description

Enable LIN mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_EnableLIN

LL_USART_DisableLIN

Function name

__STATIC_INLINE void LL_USART_DisableLIN (USART_TypeDef * USARTx)

Function description

Disable LIN mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_DisableLIN

LL_USART_IsEnabledLIN

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledLIN (USART_TypeDef * USARTx)

Function description

Indicate if LIN mode is enabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_IsEnabledLIN

LL_USART_ConfigAsyncMode

Function name

__STATIC_INLINE void LL_USART_ConfigAsyncMode (USART_TypeDef * USARTx)

Function description

Perform basic configuration of USART for enabling use in Asynchronous Mode (UART)

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- In UART mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,CLKEN bit in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register.
- Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear CLKEN in CR2 using LL_USART_DisableSCLKOutput() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() function
- Other remaining configurations items related to Asynchronous Mode (as Baud Rate, Word length, Parity, ...) should be set using dedicated functions

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_ConfigAsyncMode
- CR2 CLKEN LL_USART_ConfigAsyncMode
- CR3 SCEN LL_USART_ConfigAsyncMode
- CR3 IREN LL_USART_ConfigAsyncMode
- CR3 HDSEL LL_USART_ConfigAsyncMode

LL_USART_ConfigSyncMode

Function name

__STATIC_INLINE void LL_USART_ConfigSyncMode (USART_TypeDef * USARTx)

Function description

Perform basic configuration of USART for enabling use in Synchronous Mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- In Synchronous mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register. This function also sets the USART in Synchronous mode.
- Macro IS_USART_INSTANCE(USARTx) can be used to check whether or not Synchronous mode is supported by the USARTx instance.
- Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() functionSet CLKEN in CR2 using LL_USART_EnableSCLKOutput() function
- Other remaining configurations items related to Synchronous Mode (as Baud Rate, Word length, Parity, Clock Polarity, ...) should be set using dedicated functions

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_ConfigSyncMode
- CR2 CLKEN LL_USART_ConfigSyncMode
- CR3 SCEN LL_USART_ConfigSyncMode
- CR3 IREN LL_USART_ConfigSyncMode
- CR3 HDSEL LL_USART_ConfigSyncMode

LL_USART_ConfigLINMode

Function name

__STATIC_INLINE void LL_USART_ConfigLINMode (USART_TypeDef * USARTx)

Function description

Perform basic configuration of USART for enabling use in LIN Mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- In LIN mode, the following bits must be kept cleared: STOP and CLKEN bits in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register. This function also set the UART/USART in LIN mode.
- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.
- Call of this function is equivalent to following function call sequence : Clear CLKEN in CR2 using LL_USART_DisableSCLKOutput() functionClear STOP in CR2 using LL_USART_SetStopBitsLength() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() functionSet LINEN in CR2 using LL_USART_EnableLIN() function
- Other remaining configurations items related to LIN Mode (as Baud Rate, Word length, LIN Break Detection Length, ...) should be set using dedicated functions

Reference Manual to LL API cross reference:

- CR2 CLKEN LL_USART_ConfigLINMode
- CR2 STOP LL_USART_ConfigLINMode
- CR2 LINEN LL_USART_ConfigLINMode
- CR3 IREN LL_USART_ConfigLINMode
- CR3 SCEN LL_USART_ConfigLINMode
- CR3 HDSEL LL_USART_ConfigLINMode

LL_USART_ConfigHalfDuplexMode

Function name

__STATIC_INLINE void LL_USART_ConfigHalfDuplexMode (USART_TypeDef * USARTx)

Function description

Perform basic configuration of USART for enabling use in Half Duplex Mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- In Half Duplex mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,CLKEN bit in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register, This function also sets the UART/USART in Half Duplex mode.
- Macro IS_UART_HALFDUPLEX_INSTANCE(USARTx) can be used to check whether or not Half-Duplex mode is supported by the USARTx instance.
- Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear CLKEN in CR2 using LL_USART_DisableSCLKOutput() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear IREN in CR3 using LL_USART_DisableIrda() functionSet HDSEL in CR3 using LL_USART_EnableHalfDuplex() function
- Other remaining configurations items related to Half Duplex Mode (as Baud Rate, Word length, Parity, ...) should be set using dedicated functions

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_ConfigHalfDuplexMode
- CR2 CLKEN LL_USART_ConfigHalfDuplexMode
- CR3 HDSEL LL_USART_ConfigHalfDuplexMode
- CR3 SCEN LL_USART_ConfigHalfDuplexMode
- CR3 IREN LL_USART_ConfigHalfDuplexMode

LL_USART_ConfigSmartcardMode

Function name

__STATIC_INLINE void LL_USART_ConfigSmartcardMode (USART_TypeDef * USARTx)

Function description

Perform basic configuration of USART for enabling use in Smartcard Mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- In Smartcard mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register. This function also configures Stop bits to 1.5 bits and sets the USART in Smartcard mode (SCEN bit). Clock Output is also enabled (CLKEN).
- Macro IS_SMARTCARD_INSTANCE(USARTx) can be used to check whether or not Smartcard feature is supported by the USARTx instance.
- Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() functionConfigure STOP in CR2 using LL_USART_SetStopBitsLength() functionSet CLKEN in CR2 using LL_USART_EnableSCLKOutput() functionSet SCEN in CR3 using LL_USART_EnableSmartcard() function
- Other remaining configurations items related to Smartcard Mode (as Baud Rate, Word length, Parity, ...) should be set using dedicated functions

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_ConfigSmartcardMode
- CR2 STOP LL_USART_ConfigSmartcardMode
- CR2 CLKEN LL_USART_ConfigSmartcardMode
- CR3 HDSEL LL_USART_ConfigSmartcardMode
- CR3 SCEN LL_USART_ConfigSmartcardMode

LL_USART_ConfigIrdaMode

Function name

__STATIC_INLINE void LL_USART_ConfigIrdaMode (USART_TypeDef * USARTx)

Function description

Perform basic configuration of USART for enabling use in Irda Mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- In IRDA mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,STOP and CLKEN bits in the USART_CR2 register,SCEN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register. This function also sets the UART/USART in IRDA mode (IREN bit).
- Macro IS_IRDA_INSTANCE(USARTx) can be used to check whether or not IrDA feature is supported by the USARTx instance.
- Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear CLKEN in CR2 using LL_USART_DisableSCLKOutput() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() functionConfigure STOP in CR2 using LL_USART_SetStopBitsLength() functionSet IREN in CR3 using LL_USART_EnableIrda() function
- Other remaining configurations items related to Irda Mode (as Baud Rate, Word length, Power mode, ...) should be set using dedicated functions

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_ConfigIrdaMode
- CR2 CLKEN LL_USART_ConfigIrdaMode
- CR2 STOP LL_USART_ConfigIrdaMode
- CR3 SCEN LL_USART_ConfigIrdaMode
- CR3 HDSEL LL_USART_ConfigIrdaMode
- CR3 IREN LL_USART_ConfigIrdaMode

LL_USART_ConfigMultiProcessMode

Function name

__STATIC_INLINE void LL_USART_ConfigMultiProcessMode (USART_TypeDef * USARTx)

Function description

Perform basic configuration of USART for enabling use in Multi processor Mode (several USARTs connected in a network, one of the USARTs can be the master, its TX output connected to the RX inputs of the other slaves USARTs).

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- In MultiProcessor mode, the following bits must be kept cleared: LINEN bit in the USART_CR2 register,CLKEN bit in the USART_CR2 register,SCEN bit in the USART_CR3 register,IREN bit in the USART_CR3 register,HDSEL bit in the USART_CR3 register.
- Call of this function is equivalent to following function call sequence : Clear LINEN in CR2 using LL_USART_DisableLIN() functionClear CLKEN in CR2 using LL_USART_DisableSCLKOutput() functionClear SCEN in CR3 using LL_USART_DisableSmartcard() functionClear IREN in CR3 using LL_USART_DisableIrda() functionClear HDSEL in CR3 using LL_USART_DisableHalfDuplex() function
- Other remaining configurations items related to Multi processor Mode (as Baud Rate, Wake Up Method, Node address, ...) should be set using dedicated functions

Reference Manual to LL API cross reference:

- CR2 LINEN LL_USART_ConfigMultiProcessMode
- CR2 CLKEN LL_USART_ConfigMultiProcessMode
- CR3 SCEN LL_USART_ConfigMultiProcessMode
- CR3 HDSEL LL_USART_ConfigMultiProcessMode
- CR3 IREN LL_USART_ConfigMultiProcessMode

LL_USART_IsActiveFlag_PE

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_PE (USART_TypeDef * USARTx)

Function description

Check if the USART Parity Error Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR PE LL_USART_IsActiveFlag_PE

LL_USART_IsActiveFlag_FE

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_FE (USART_TypeDef * USARTx)

Function description

Check if the USART Framing Error Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR FE LL_USART_IsActiveFlag_FE

LL_USART_IsActiveFlag_NE

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_NE (USART_TypeDef * USARTx)

Function description

Check if the USART Noise error detected Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR NF LL_USART_IsActiveFlag_NE

LL_USART_IsActiveFlag_ORE

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_ORE (USART_TypeDef * USARTx)

Function description

Check if the USART OverRun Error Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR ORE LL_USART_IsActiveFlag_ORE

LL_USART_IsActiveFlag_IDLE

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_IDLE (USART_TypeDef * USARTx)

Function description

Check if the USART IDLE line detected Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR IDLE LL_USART_IsActiveFlag_IDLE

LL_USART_IsActiveFlag_RXNE

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_RXNE (USART_TypeDef * USARTx)

Function description

Check if the USART Read Data Register Not Empty Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR RXNE LL_USART_IsActiveFlag_RXNE

LL_USART_IsActiveFlag_TC

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_TC (USART_TypeDef * USARTx)

Function description

Check if the USART Transmission Complete Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR TC LL_USART_IsActiveFlag_TC

LL_USART_IsActiveFlag_TXE

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_TXE (USART_TypeDef * USARTx)

Function description

Check if the USART Transmit Data Register Empty Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- SR TXE LL_USART_IsActiveFlag_TXE

LL_USART_IsActiveFlag_LBD

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_LBD (USART_TypeDef * USARTx)

Function description

Check if the USART LIN Break Detection Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- SR LBD LL_USART_IsActiveFlag_LBD

LL_USART_IsActiveFlag_nCTS

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_nCTS (USART_TypeDef * USARTx)

Function description

Check if the USART CTS Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- SR CTS LL_USART_IsActiveFlag_nCTS

LL_USART_IsActiveFlag_SBK

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_SBK (USART_TypeDef * USARTx)

Function description

Check if the USART Send Break Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 SBK LL_USART_IsActiveFlag_SBK

LL_USART_IsActiveFlag_RWU

Function name

__STATIC_INLINE uint32_t LL_USART_IsActiveFlag_RWU (USART_TypeDef * USARTx)

Function description

Check if the USART Receive Wake Up from mute mode Flag is set or not.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 RWU LL_USART_IsActiveFlag_RWU

LL_USART_ClearFlag_PE

Function name

__STATIC_INLINE void LL_USART_ClearFlag_PE (USART_TypeDef * USARTx)

Function description

Clear Parity Error Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the USARTx_SR register followed by a read access to the USARTx_DR register.
- Please also consider that when clearing this flag, other flags as NE, FE, ORE, IDLE would also be cleared.

Reference Manual to LL API cross reference:

- SR PE LL_USART_ClearFlag_PE

LL_USART_ClearFlag_FE

Function name

__STATIC_INLINE void LL_USART_ClearFlag_FE (USART_TypeDef * USARTx)

Function description

Clear Framing Error Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the USARTx_SR register followed by a read access to the USARTx_DR register.
- Please also consider that when clearing this flag, other flags as PE, NE, ORE, IDLE would also be cleared.

Reference Manual to LL API cross reference:

- SR FE LL_USART_ClearFlag_FE

LL_USART_ClearFlag_NE

Function name

__STATIC_INLINE void LL_USART_ClearFlag_NE (USART_TypeDef * USARTx)

Function description

Clear Noise detected Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the USARTx_SR register followed by a read access to the USARTx_DR register.
- Please also consider that when clearing this flag, other flags as PE, FE, ORE, IDLE would also be cleared.

Reference Manual to LL API cross reference:

- SR NF LL_USART_ClearFlag_NE

LL_USART_ClearFlag_ORE

Function name

__STATIC_INLINE void LL_USART_ClearFlag_ORE (USART_TypeDef * USARTx)

Function description

Clear OverRun Error Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the USARTx_SR register followed by a read access to the USARTx_DR register.
- Please also consider that when clearing this flag, other flags as PE, NE, FE, IDLE would also be cleared.

Reference Manual to LL API cross reference:

- SR ORE LL_USART_ClearFlag_ORE

LL_USART_ClearFlag_IDLE

Function name

__STATIC_INLINE void LL_USART_ClearFlag_IDLE (USART_TypeDef * USARTx)

Function description

Clear IDLE line detected Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Clearing this flag is done by a read access to the USARTx_SR register followed by a read access to the USARTx_DR register.
- Please also consider that when clearing this flag, other flags as PE, NE, FE, ORE would also be cleared.

Reference Manual to LL API cross reference:

- SR IDLE LL_USART_ClearFlag_IDLE

LL_USART_ClearFlag_TC

Function name

__STATIC_INLINE void LL_USART_ClearFlag_TC (USART_TypeDef * USARTx)

Function description

Clear Transmission Complete Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR TC LL_USART_ClearFlag_TC

LL_USART_ClearFlag_RXNE

Function name

__STATIC_INLINE void LL_USART_ClearFlag_RXNE (USART_TypeDef * USARTx)

Function description

Clear RX Not Empty Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR RXNE LL_USART_ClearFlag_RXNE

LL_USART_ClearFlag_LBD

Function name

__STATIC_INLINE void LL_USART_ClearFlag_LBD (USART_TypeDef * USARTx)

Function description

Clear LIN Break Detection Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- SR LBD LL_USART_ClearFlag_LBD

LL_USART_ClearFlag_nCTS

Function name

__STATIC_INLINE void LL_USART_ClearFlag_nCTS (USART_TypeDef * USARTx)

Function description

Clear CTS Interrupt Flag.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- SR CTS LL_USART_ClearFlag_nCTS

LL_USART_EnableIT_IDLE

Function name

__STATIC_INLINE void LL_USART_EnableIT_IDLE (USART_TypeDef * USARTx)

Function description

Enable IDLE Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 IDLEIE LL_USART_EnableIT_IDLE

LL_USART_EnableIT_RXNE

Function name

__STATIC_INLINE void LL_USART_EnableIT_RXNE (USART_TypeDef * USARTx)

Function description

Enable RX Not Empty Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RXNEIE LL_USART_EnableIT_RXNE

LL_USART_EnableIT_TC

Function name

__STATIC_INLINE void LL_USART_EnableIT_TC (USART_TypeDef * USARTx)

Function description

Enable Transmission Complete Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TCIE LL_USART_EnableIT_TC

LL_USART_EnableIT_TXE

Function name

__STATIC_INLINE void LL_USART_EnableIT_TXE (USART_TypeDef * USARTx)

Function description

Enable TX Empty Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TXEIE LL_USART_EnableIT_TXE

LL_USART_EnableIT_PE

Function name

__STATIC_INLINE void LL_USART_EnableIT_PE (USART_TypeDef * USARTx)

Function description

Enable Parity Error Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 PEIE LL_USART_EnableIT_PE

LL_USART_EnableIT_LBD

Function name

__STATIC_INLINE void LL_USART_EnableIT_LBD (USART_TypeDef * USARTx)

Function description

Enable LIN Break Detection Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LBDIE LL_USART_EnableIT_LBD

LL_USART_EnableIT_ERROR

Function name

__STATIC_INLINE void LL_USART_EnableIT_ERROR (USART_TypeDef * USARTx)

Function description

Enable Error Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- When set, Error Interrupt Enable Bit is enabling interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the USARTx_SR register). 0: Interrupt is inhibited 1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the USARTx_SR register.

Reference Manual to LL API cross reference:

- CR3 EIE LL_USART_EnableIT_ERROR

LL_USART_EnableIT_CTS

Function name

__STATIC_INLINE void LL_USART_EnableIT_CTS (USART_TypeDef * USARTx)

Function description

Enable CTS Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 CTSIE LL_USART_EnableIT_CTS

LL_USART_DisableIT_IDLE

Function name

__STATIC_INLINE void LL_USART_DisableIT_IDLE (USART_TypeDef * USARTx)

Function description

Disable IDLE Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 IDLEIE LL_USART_DisableIT_IDLE

LL_USART_DisableIT_RXNE

Function name

__STATIC_INLINE void LL_USART_DisableIT_RXNE (USART_TypeDef * USARTx)

Function description

Disable RX Not Empty Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RXNEIE LL_USART_DisableIT_RXNE

LL_USART_DisableIT_TC

Function name

__STATIC_INLINE void LL_USART_DisableIT_TC (USART_TypeDef * USARTx)

Function description

Disable Transmission Complete Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TCIE LL_USART_DisableIT_TC

LL_USART_DisableIT_TXE

Function name

__STATIC_INLINE void LL_USART_DisableIT_TXE (USART_TypeDef * USARTx)

Function description

Disable TX Empty Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 TXEIE LL_USART_DisableIT_TXE

LL_USART_DisableIT_PE

Function name

__STATIC_INLINE void LL_USART_DisableIT_PE (USART_TypeDef * USARTx)

Function description

Disable Parity Error Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 PEIE LL_USART_DisableIT_PE

LL_USART_DisableIT_LBD

Function name

__STATIC_INLINE void LL_USART_DisableIT_LBD (USART_TypeDef * USARTx)

Function description

Disable LIN Break Detection Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LBDIE LL_USART_DisableIT_LBD

LL_USART_DisableIT_ERROR

Function name

__STATIC_INLINE void LL_USART_DisableIT_ERROR (USART_TypeDef * USARTx)

Function description

Disable Error Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- When set, Error Interrupt Enable Bit is enabling interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the USARTx_SR register). 0: Interrupt is inhibited 1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the USARTx_SR register.

Reference Manual to LL API cross reference:

- CR3 EIE LL_USART_DisableIT_ERROR

LL_USART_DisableIT_CTS

Function name

__STATIC_INLINE void LL_USART_DisableIT_CTS (USART_TypeDef * USARTx)

Function description

Disable CTS Interrupt.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 CTSIE LL_USART_DisableIT_CTS

LL_USART_IsEnabledIT_IDLE

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_IDLE (USART_TypeDef * USARTx)

Function description

Check if the USART IDLE Interrupt source is enabled or disabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 IDLEIE LL_USART_IsEnabledIT_IDLE

LL_USART_IsEnabledIT_RXNE

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_RXNE (USART_TypeDef * USARTx)

Function description

Check if the USART RX Not Empty Interrupt is enabled or disabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 RXNEIE LL_USART_IsEnabledIT_RXNE

LL_USART_IsEnabledIT_TC

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_TC (USART_TypeDef * USARTx)

Function description

Check if the USART Transmission Complete Interrupt is enabled or disabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 TCIE LL_USART_IsEnabledIT_TC

LL_USART_IsEnabledIT_TXE

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_TXE (USART_TypeDef * USARTx)

Function description

Check if the USART TX Empty Interrupt is enabled or disabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 TXEIE LL_USART_IsEnabledIT_TXE

LL_USART_IsEnabledIT_PE

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_PE (USART_TypeDef * USARTx)

Function description

Check if the USART Parity Error Interrupt is enabled or disabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR1 PEIE LL_USART_IsEnabledIT_PE

LL_USART_IsEnabledIT_LBD

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_LBD (USART_TypeDef * USARTx)

Function description

Check if the USART LIN Break Detection Interrupt is enabled or disabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_UART_LIN_INSTANCE(USARTx) can be used to check whether or not LIN feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR2 LBDIE LL_USART_IsEnabledIT_LBD

LL_USART_IsEnabledIT_ERROR

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_ERROR (USART_TypeDef * USARTx)

Function description

Check if the USART Error Interrupt is enabled or disabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR3 EIE LL_USART_IsEnabledIT_ERROR

LL_USART_IsEnabledIT_CTS

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledIT_CTS (USART_TypeDef * USARTx)

Function description

Check if the USART CTS Interrupt is enabled or disabled.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Notes

- Macro IS_UART_HWFLOW_INSTANCE(USARTx) can be used to check whether or not Hardware Flow control feature is supported by the USARTx instance.

Reference Manual to LL API cross reference:

- CR3 CTSIE LL_USART_IsEnabledIT_CTS

LL_USART_EnableDMAReq_RX

Function name

__STATIC_INLINE void LL_USART_EnableDMAReq_RX (USART_TypeDef * USARTx)

Function description

Enable DMA Mode for reception.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR3 DMAR LL_USART_EnableDMAReq_RX

LL_USART_DisableDMAReq_RX

Function name

__STATIC_INLINE void LL_USART_DisableDMAReq_RX (USART_TypeDef * USARTx)

Function description

Disable DMA Mode for reception.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR3 DMAR LL_USART_DisableDMAReq_RX

LL_USART_IsEnabledDMAReq_RX

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledDMAReq_RX (USART_TypeDef * USARTx)

Function description

Check if DMA Mode is enabled for reception.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR3 DMAR LL_USART_IsEnabledDMAReq_RX

LL_USART_EnableDMAReq_TX

Function name

__STATIC_INLINE void LL_USART_EnableDMAReq_TX (USART_TypeDef * USARTx)

Function description

Enable DMA Mode for transmission.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR3 DMAT LL_USART_EnableDMAReq_TX

LL_USART_DisableDMAReq_TX

Function name

__STATIC_INLINE void LL_USART_DisableDMAReq_TX (USART_TypeDef * USARTx)

Function description

Disable DMA Mode for transmission.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR3 DMAT LL_USART_DisableDMAReq_TX

LL_USART_IsEnabledDMAReq_TX

Function name

__STATIC_INLINE uint32_t LL_USART_IsEnabledDMAReq_TX (USART_TypeDef * USARTx)

Function description

Check if DMA Mode is enabled for transmission.

Parameters

- **USARTx:** USART Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR3 DMAT LL_USART_IsEnabledDMAReq_TX

LL_USART_DMA_GetRegAddr

Function name

__STATIC_INLINE uint32_t LL_USART_DMA_GetRegAddr (USART_TypeDef * USARTx)

Function description

Get the data register address used for DMA transfer.

Parameters

- **USARTx:** USART Instance

Return values

- **Address:** of data register

Notes

- Address of Data Register is valid for both Transmit and Receive transfers.

Reference Manual to LL API cross reference:

- DR DR LL_USART_DMA_GetRegAddr

LL_USART_ReceiveData8

Function name

__STATIC_INLINE uint8_t LL_USART_ReceiveData8 (USART_TypeDef * USARTx)

Function description

Read Receiver Data register (Receive Data value, 8 bits)

Parameters

- **USARTx:** USART Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Reference Manual to LL API cross reference:

- DR DR LL_USART_ReceiveData8

LL_USART_ReceiveData9

Function name

__STATIC_INLINE uint16_t LL_USART_ReceiveData9 (USART_TypeDef * USARTx)

Function description

Read Receiver Data register (Receive Data value, 9 bits)

Parameters

- **USARTx:** USART Instance

Return values

- **Value:** between Min_Data=0x00 and Max_Data=0x1FF

Reference Manual to LL API cross reference:

- DR DR LL_USART_ReceiveData9

LL_USART_TransmitData8

Function name

__STATIC_INLINE void LL_USART_TransmitData8 (USART_TypeDef * USARTx, uint8_t Value)

Function description

Write in Transmitter Data Register (Transmit Data value, 8 bits)

Parameters

- **USARTx:** USART Instance
- **Value:** between Min_Data=0x00 and Max_Data=0xFF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR DR LL_USART_TransmitData8

LL_USART_TransmitData9

Function name

__STATIC_INLINE void LL_USART_TransmitData9 (USART_TypeDef * USARTx, uint16_t Value)

Function description

Write in Transmitter Data Register (Transmit Data value, 9 bits)

Parameters

- **USARTx:** USART Instance
- **Value:** between Min_Data=0x00 and Max_Data=0x1FF

Return values

- **None:**

Reference Manual to LL API cross reference:

- DR DR LL_USART_TransmitData9

LL_USART_RequestBreakSending

Function name

__STATIC_INLINE void LL_USART_RequestBreakSending (USART_TypeDef * USARTx)

Function description

Request Break sending.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 SBK LL_USART_RequestBreakSending

LL_USART_RequestEnterMuteMode

Function name

__STATIC_INLINE void LL_USART_RequestEnterMuteMode (USART_TypeDef * USARTx)

Function description

Put USART in Mute mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RWU LL_USART_RequestEnterMuteMode

LL_USART_RequestExitMuteMode

Function name

__STATIC_INLINE void LL_USART_RequestExitMuteMode (USART_TypeDef * USARTx)

Function description

Put USART in Active mode.

Parameters

- **USARTx:** USART Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- CR1 RWU LL_USART_RequestExitMuteMode

LL_USART_DeInit

Function name

ErrorStatus LL_USART_DeInit (USART_TypeDef * USARTx)

Function description

De-initialize USART registers (Registers restored to their default values).

Parameters

- **USARTx:** USART Instance

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: USART registers are de-initialized
  - ERROR: USART registers are not de-initialized

LL_USART_Init

Function name

ErrorStatus LL_USART_Init (USART_TypeDef * USARTx, LL_USART_InitTypeDef * USART_InitStruct)

Function description

Initialize USART registers according to the specified parameters in USART_InitStruct.

Parameters

- **USARTx:** USART Instance
- **USART_InitStruct:** pointer to a LL_USART_InitTypeDef structure that contains the configuration information for the specified USART peripheral.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: USART registers are initialized according to USART_InitStruct content
  - ERROR: Problem occurred during USART Registers initialization

Notes

- As some bits in USART configuration registers can only be written when the USART is disabled (USART_CR1_UE bit =0), USART IP should be in disabled state prior calling this function. Otherwise, ERROR result will be returned.
- Baud rate value stored in USART_InitStruct BaudRate field, should be valid (different from 0).

LL_USART_StructInit

Function name

void LL_USART_StructInit (LL_USART_InitTypeDef * USART_InitStruct)

Function description

Set each LL_USART_InitTypeDef field to default value.

Parameters

- **USART_InitStruct:** Pointer to a LL_USART_InitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

LL_USART_ClockInit

Function name

ErrorStatus LL_USART_ClockInit (USART_TypeDef * USARTx, LL_USART_ClockInitTypeDef * USART_ClockInitStruct)

Function description

Initialize USART Clock related settings according to the specified parameters in the USART_ClockInitStruct.

Parameters

- **USARTx:** USART Instance
- **USART_ClockInitStruct:** Pointer to a LL_USART_ClockInitTypeDef structure that contains the Clock configuration information for the specified USART peripheral.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: USART registers related to Clock settings are initialized according to USART_ClockInitStruct content
  - ERROR: Problem occurred during USART Registers initialization

Notes

- As some bits in USART configuration registers can only be written when the USART is disabled (USART_CR1_UE bit =0), USART IP should be in disabled state prior calling this function. Otherwise, ERROR result will be returned.

LL_USART_ClockStructInit

Function name

void LL_USART_ClockStructInit (LL_USART_ClockInitTypeDef * USART_ClockInitStruct)

Function description

Set each field of a LL_USART_ClockInitTypeDef type structure to default value.

Parameters

- **USART_ClockInitStruct:** Pointer to a LL_USART_ClockInitTypeDef structure whose fields will be set to default values.

Return values

- **None:**

93.3 USART Firmware driver defines

The following section lists the various define and macros of the module.

93.3.1 USART

USART

Clock Signal

LL_USART_CLOCK_DISABLE

Clock signal not provided

LL_USART_CLOCK_ENABLE

Clock signal provided

Datawidth

LL_USART_DATAWIDTH_8B

8 bits word length : Start bit, 8 data bits, n stop bits

LL_USART_DATAWIDTH_9B

9 bits word length : Start bit, 9 data bits, n stop bits

Communication Direction

LL_USART_DIRECTION_NONE

Transmitter and Receiver are disabled

LL_USART_DIRECTION_RX

Transmitter is disabled and Receiver is enabled

LL_USART_DIRECTION_TX

Transmitter is enabled and Receiver is disabled

LL_USART_DIRECTION_TX_RX

Transmitter and Receiver are enabled

Get Flags Defines

LL_USART_SR_PE

Parity error flag

LL_USART_SR_FE

Framing error flag

LL_USART_SR_NE

Noise detected flag

LL_USART_SR_ORE

Overrun error flag

LL_USART_SR_IDLE

Idle line detected flag

LL_USART_SR_RXNE

Read data register not empty flag

LL_USART_SR_TC

Transmission complete flag

LL_USART_SR_TXE

Transmit data register empty flag

LL_USART_SR_LBD

LIN break detection flag

LL_USART_SR_CTS

CTS flag

Hardware Control

LL_USART_HWCONTROL_NONE

CTS and RTS hardware flow control disabled

LL_USART_HWCONTROL_RTS

RTS output enabled, data is only requested when there is space in the receive buffer

LL_USART_HWCONTROL_CTS

CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0)

LL_USART_HWCONTROL_RTS_CTS

CTS and RTS hardware flow control enabled

IrDA Power

LL_USART_IRDA_POWER_NORMAL

IrDA normal power mode

LL_USART_IRDA_POWER_LOW

IrDA low power mode

IT Defines

LL_USART_CR1_IDLEIE

IDLE interrupt enable

LL_USART_CR1_RXNEIE

Read data register not empty interrupt enable

LL_USART_CR1_TCIE

Transmission complete interrupt enable

LL_USART_CR1_TXEIE

Transmit data register empty interrupt enable

LL_USART_CR1_PEIE

Parity error

LL_USART_CR2_LBDIE

LIN break detection interrupt enable

LL_USART_CR3_EIE

Error interrupt enable

LL_USART_CR3_CTSIE

CTS interrupt enable

Last Clock Pulse

LL_USART_LASTCLKPULSE_NO_OUTPUT

The clock pulse of the last data bit is not output to the SCLK pin

LL_USART_LASTCLKPULSE_OUTPUT

The clock pulse of the last data bit is output to the SCLK pin

LIN Break Detection Length

LL_USART_LINBREAK_DETECT_10B

10-bit break detection method selected

LL_USART_LINBREAK_DETECT_11B

11-bit break detection method selected

Oversampling

LL_USART_OVERSAMPLING_16

Oversampling by 16

LL_USART_OVERSAMPLING_8

Oversampling by 8

Parity Control

LL_USART_PARITY_NONE

Parity control disabled

LL_USART_PARITY_EVEN

Parity control enabled and Even Parity is selected

LL_USART_PARITY_ODD

Parity control enabled and Odd Parity is selected

Clock Phase

LL_USART_PHASE_1EDGE

The first clock transition is the first data capture edge

LL_USART_PHASE_2EDGE

The second clock transition is the first data capture edge

Clock Polarity

LL_USART_POLARITY_LOW

Steady low value on SCLK pin outside transmission window

LL_USART_POLARITY_HIGH

Steady high value on SCLK pin outside transmission window

Stop Bits

LL_USART_STOPBITS_0_5

0.5 stop bit

LL_USART_STOPBITS_1

1 stop bit

LL_USART_STOPBITS_1_5

1.5 stop bits

LL_USART_STOPBITS_2

2 stop bits

Wakeup

LL_USART_WAKEUP_IDLELINE

USART wake up from Mute mode on Idle Line

LL_USART_WAKEUP_ADDRESSMARK

USART wake up from Mute mode on Address Mark

Exported_Macros_Helper

__LL_USART_DIV_SAMPLING8_100

Description:

- Compute USARTDIV value according to Peripheral Clock and expected Baud Rate in 8 bits sampling mode (32 bits value of USARTDIV is returned)

Parameters:

- __PERIPHCLK__: Peripheral Clock frequency used for USART instance
- __BAUDRATE__: Baud rate value to achieve

Return value:

- USARTDIV: value to be used for BRR register filling in OverSampling_8 case

__LL_USART_DIVMANT_SAMPLING8

__LL_USART_DIVFRAQ_SAMPLING8

__LL_USART_DIV_SAMPLING8

__LL_USART_DIV_SAMPLING16_100

Description:

- Compute USARTDIV value according to Peripheral Clock and expected Baud Rate in 16 bits sampling mode (32 bits value of USARTDIV is returned)

Parameters:

- __PERIPHCLK__: Peripheral Clock frequency used for USART instance
- __BAUDRATE__: Baud rate value to achieve

Return value:

- USARTDIV: value to be used for BRR register filling in OverSampling_16 case

__LL_USART_DIVMANT_SAMPLING16

__LL_USART_DIVFRAQ_SAMPLING16

__LL_USART_DIV_SAMPLING16

Common Write and read registers Macros

LL_USART_WriteReg

Description:

- Write a value in USART register.

Parameters:

- __INSTANCE__: USART Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_USART_ReadReg

Description:

- Read a value in USART register.

Parameters:

- __INSTANCE__: USART Instance
- __REG__: Register to be read

Return value:

- Register: value

94 LL UTILS Generic Driver

94.1 UTILS Firmware driver registers structures

94.1.1 LL_UTILS_PLLInitTypeDef

LL_UTILS_PLLInitTypeDef is defined in the stm32f4xx_ll_utils.h

Data Fields

- ***uint32_t PLLM***
- ***uint32_t PLLN***
- ***uint32_t PLLP***

Field Documentation

- ***uint32_t LL_UTILS_PLLInitTypeDef::PLLM***

  Division factor for PLL VCO input clock. This parameter can be a value of ***RCC_LL_EC_PLLM_DIV***This feature can be modified afterwards using unitary function **LL_RCC_PLL_ConfigDomain_SYS()**.
- ***uint32_t LL_UTILS_PLLInitTypeDef::PLLN***

  Multiplication factor for PLL VCO output clock. This parameter must be a number between Min_Data = RCC_PLLN_MIN_VALUE and Max_Data = RCC_PLLN_MIN_VALUEThis feature can be modified afterwards using unitary function **LL_RCC_PLL_ConfigDomain_SYS()**.
- ***uint32_t LL_UTILS_PLLInitTypeDef::PLLP***

  Division for the main system clock. This parameter can be a value of ***RCC_LL_EC_PLLP_DIV***This feature can be modified afterwards using unitary function **LL_RCC_PLL_ConfigDomain_SYS()**.

94.1.2 LL_UTILS_ClkInitTypeDef

LL_UTILS_ClkInitTypeDef is defined in the stm32f4xx_ll_utils.h

Data Fields

- ***uint32_t AHBCLKDivider***
- ***uint32_t APB1CLKDivider***
- ***uint32_t APB2CLKDivider***

Field Documentation

- ***uint32_t LL_UTILS_ClkInitTypeDef::AHBCLKDivider***

  The AHB clock (HCLK) divider. This clock is derived from the system clock (SYSCLK). This parameter can be a value of ***RCC_LL_EC_SYSCLK_DIV***This feature can be modified afterwards using unitary function **LL_RCC_SetAHBPrescaler()**.
- ***uint32_t LL_UTILS_ClkInitTypeDef::APB1CLKDivider***

  The APB1 clock (PCLK1) divider. This clock is derived from the AHB clock (HCLK). This parameter can be a value of ***RCC_LL_EC_APB1_DIV***This feature can be modified afterwards using unitary function **LL_RCC_SetAPB1Prescaler()**.
- ***uint32_t LL_UTILS_ClkInitTypeDef::APB2CLKDivider***

  The APB2 clock (PCLK2) divider. This clock is derived from the AHB clock (HCLK). This parameter can be a value of ***RCC_LL_EC_APB2_DIV***This feature can be modified afterwards using unitary function **LL_RCC_SetAPB2Prescaler()**.

94.2 UTILS Firmware driver API description

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

94.2.1 System Configuration functions

System, AHB and APB buses clocks configuration

- The maximum frequency of the SYSCLK, HCLK, PCLK1 and PCLK2 is 180000000 Hz.

This section contains the following APIs:

- ***LL_SetSystemCoreClock()***

- *LL_PLL_ConfigSystemClock_HSI()*
- *LL_PLL_ConfigSystemClock_HSE()*

94.2.2 Detailed description of functions

LL_GetUID_Word0

Function name

__STATIC_INLINE uint32_t LL_GetUID_Word0 (void)

Function description

Get Word0 of the unique device identifier (UID based on 96 bits)

Return values

- **UID[31:0]:**

LL_GetUID_Word1

Function name

__STATIC_INLINE uint32_t LL_GetUID_Word1 (void)

Function description

Get Word1 of the unique device identifier (UID based on 96 bits)

Return values

- **UID[63:32]:**

LL_GetUID_Word2

Function name

__STATIC_INLINE uint32_t LL_GetUID_Word2 (void)

Function description

Get Word2 of the unique device identifier (UID based on 96 bits)

Return values

- **UID[95:64]:**

LL_GetFlashSize

Function name

__STATIC_INLINE uint32_t LL_GetFlashSize (void)

Function description

Get Flash memory size.

Return values

- **FLASH_SIZE[15:0]:** Flash memory size

Notes

- This bitfield indicates the size of the device Flash memory expressed in Kbytes. As an example, 0x040 corresponds to 64 Kbytes.

LL_GetPackageType

Function name

__STATIC_INLINE uint32_t LL_GetPackageType (void)

Function description

Get Package type.

Return values

- **Returned:** value can be one of the following values:
  - LL_UTILS_PACKAGETYPE_WLCSP36_UFQFPN48_LQFP64 (*)
  - LL_UTILS_PACKAGETYPE_WLCSP168_FBGA169_LQFP100_LQFP64_UFQFPN48 (*)
  - LL_UTILS_PACKAGETYPE_WLCSP64_WLCSP81_LQFP176_UFBGA176 (*)
  - LL_UTILS_PACKAGETYPE_LQFP144_UFBGA144_UFBGA144_UFBGA100 (*)
  - LL_UTILS_PACKAGETYPE_LQFP100_LQFP208_TFBGA216 (*)
  - LL_UTILS_PACKAGETYPE_LQFP208_TFBGA216 (*)
  - LL_UTILS_PACKAGETYPE_TQFP64_UFBGA144_LQFP144 (*)

  (*) value not defined in all devices.

LL_InitTick

Function name

__STATIC_INLINE void LL_InitTick (uint32_t HCLKFrequency, uint32_t Ticks)

Function description

This function configures the Cortex-M SysTick source of the time base.

Parameters

- **HCLKFrequency:** HCLK frequency in Hz (can be calculated thanks to RCC helper macro)
- **Ticks:** Number of ticks

Return values

- **None:**

Notes

- When a RTOS is used, it is recommended to avoid changing the SysTick configuration by calling this function, for a delay use rather osDelay RTOS service.

LL_Init1msTick

Function name

void LL_Init1msTick (uint32_t HCLKFrequency)

Function description

This function configures the Cortex-M SysTick source to have 1ms time base.

Parameters

- **HCLKFrequency:** HCLK frequency in Hz

Return values

- **None:**

Notes

- When a RTOS is used, it is recommended to avoid changing the Systick configuration by calling this function, for a delay use rather osDelay RTOS service.
- HCLK frequency can be calculated thanks to RCC helper macro or function LL_RCC_GetSystemClocksFreq

LL_mDelay

Function name

void LL_mDelay (uint32_t Delay)

Function description

This function provides accurate delay (in milliseconds) based on SysTick counter flag.

Parameters

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

Return values

- **None:**

Notes

- When a RTOS is used, it is recommended to avoid using blocking delay and use rather osDelay service.
- To respect 1ms timebase, user should call LL_Init1msTick function which will configure Systick to 1ms

LL_SetSystemCoreClock

Function name

void LL_SetSystemCoreClock (uint32_t HCLKFrequency)

Function description

This function sets directly SystemCoreClock CMSIS variable.

Parameters

- **HCLKFrequency:** HCLK frequency in Hz (can be calculated thanks to RCC helper macro)

Return values

- **None:**

Notes

- Variable can be calculated also through SystemCoreClockUpdate function.

LL_PLL_ConfigSystemClock_HSI

Function name

ErrorStatus LL_PLL_ConfigSystemClock_HSI (LL_UTILS_PLLInitTypeDef * UTILS_PLLInitStruct, LL_UTILS_ClkInitTypeDef * UTILS_ClkInitStruct)

Function description

This function configures system clock at maximum frequency with HSI as clock source of the PLL.

Parameters

- **UTILS_PLLInitStruct:** pointer to a LL_UTILS_PLLInitTypeDef structure that contains the configuration information for the PLL.
- **UTILS_ClkInitStruct:** pointer to a LL_UTILS_ClkInitTypeDef structure that contains the configuration information for the BUS prescalers.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: Max frequency configuration done
  - ERROR: Max frequency configuration not done

Notes

- The application need to ensure that PLL is disabled.
- Function is based on the following formula: PLL output frequency = (((HSI frequency / PLLM) * PLLN) / PLLP)PLLM: ensure that the VCO input frequency ranges from RCC_PLLVCO_INPUT_MIN to RCC_PLLVCO_INPUT_MAX (PLLVCO_input = HSI frequency / PLLM)PLLN: ensure that the VCO output frequency is between RCC_PLLVCO_OUTPUT_MIN and RCC_PLLVCO_OUTPUT_MAX (PLLVCO_output = PLLVCO_input * PLLN)PLLP: ensure that max frequency at 180000000 Hz is reach (PLLVCO_output / PLLP)

LL_PLL_ConfigSystemClock_HSE

Function name

ErrorStatus LL_PLL_ConfigSystemClock_HSE (uint32_t HSEFrequency, uint32_t HSEBypass, LL_UTILS_PLLInitTypeDef * UTILS_PLLInitStruct, LL_UTILS_ClkInitTypeDef * UTILS_ClkInitStruct)

Function description

This function configures system clock with HSE as clock source of the PLL.

Parameters

- **HSEFrequency:** Value between Min_Data = 4000000 and Max_Data = 26000000
- **HSEBypass:** This parameter can be one of the following values:
  - LL_UTILS_HSEBYPASS_ON
  - LL_UTILS_HSEBYPASS_OFF
- **UTILS_PLLInitStruct:** pointer to a LL_UTILS_PLLInitTypeDef structure that contains the configuration information for the PLL.
- **UTILS_ClkInitStruct:** pointer to a LL_UTILS_ClkInitTypeDef structure that contains the configuration information for the BUS prescalers.

Return values

- **An:** ErrorStatus enumeration value:
  - SUCCESS: Max frequency configuration done
  - ERROR: Max frequency configuration not done

Notes

- The application need to ensure that PLL is disabled. PLL output frequency = (((HSI frequency / PLLM) * PLLN) / PLLP)PLLM: ensure that the VCO input frequency ranges from RCC_PLLVCO_INPUT_MIN to RCC_PLLVCO_INPUT_MAX (PLLVCO_input = HSI frequency / PLLM)PLLN: ensure that the VCO output frequency is between RCC_PLLVCO_OUTPUT_MIN and RCC_PLLVCO_OUTPUT_MAX (PLLVCO_output = PLLVCO_input * PLLN)PLLP: ensure that max frequency at 180000000 Hz is reach (PLLVCO_output / PLLP)

94.3 UTILS Firmware driver defines

The following section lists the various define and macros of the module.

94.3.1 UTILS

UTILS

HSE Bypass activation

LL_UTILS_HSEBYPASS_OFF

HSE Bypass is not enabled

LL_UTILS_HSEBYPASS_ON

HSE Bypass is enabled

PACKAGE TYPE

LL_UTILS_PACKAGETYPE_WLCSP36_UFQFPN48_LQFP64

WLCSP36 or UFQFPN48 or LQFP64 package type

LL_UTILS_PACKAGETYPE_WLCSP168_FBGA169_LQFP100_LQFP64_UFQFPN48

WLCSP168 or FBGA169 or LQFP100 or LQFP64 or UFQFPN48 package type

LL_UTILS_PACKAGETYPE_WLCSP64_WLCSP81_LQFP176_UFBGA176

WLCSP64 or WLCSP81 or LQFP176 or UFBGA176 package type

LL_UTILS_PACKAGETYPE_LQFP144_UFBGA144_UFBGA144_UFBGA100

LQFP144 or UFBGA144 or UFBGA144 or UFBGA100 package type

LL_UTILS_PACKAGETYPE_LQFP100_LQFP208_TFBGA216

LQFP100 or LQFP208 or TFBGA216 package type

LL_UTILS_PACKAGETYPE_LQFP208_TFBGA216

LQFP208 or TFBGA216 package type

LL_UTILS_PACKAGETYPE_TQFP64_UFBGA144_LQFP144

TQFP64 or UFBGA144 or LQFP144 package type

95 LL WWDG Generic Driver

95.1 WWDG Firmware driver API description

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

95.1.1 Detailed description of functions

LL_WWDG_Enable

Function name

__STATIC_INLINE void LL_WWDG_Enable (WWDG_TypeDef * WWDGx)

Function description

Enable Window Watchdog.

Parameters

- **WWDGx:** WWDG Instance

Return values

- **None:**

Notes

- It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset. This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

Reference Manual to LL API cross reference:

- CR WDGA LL_WWDG_Enable

LL_WWDG_IsEnabled

Function name

__STATIC_INLINE uint32_t LL_WWDG_IsEnabled (WWDG_TypeDef * WWDGx)

Function description

Checks if Window Watchdog is enabled.

Parameters

- **WWDGx:** WWDG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CR WDGA LL_WWDG_IsEnabled

LL_WWDG_SetCounter

Function name

__STATIC_INLINE void LL_WWDG_SetCounter (WWDG_TypeDef * WWDGx, uint32_t Counter)

Function description

Set the Watchdog counter value to provided value (7-bits T[6:0])

Parameters

- **WWDGx:** WWDG Instance
- **Counter:** 0..0x7F (7 bit counter value)

Return values

- **None:**

Notes

- When writing to the WWDG_CR register, always write 1 in the MSB b6 to avoid generating an immediate reset This counter is decremented every (4096 x 2expWDGTB) PCLK cycles A reset is produced when it rolls over from 0x40 to 0x3F (bit T6 becomes cleared) Setting the counter lower then 0x40 causes an immediate reset (if WWDG enabled)

Reference Manual to LL API cross reference:

- CR T LL_WWDG_SetCounter

LL_WWDG_GetCounter

Function name

__STATIC_INLINE uint32_t LL_WWDG_GetCounter (WWDG_TypeDef * WWDGx)

Function description

Return current Watchdog Counter Value (7 bits counter value)

Parameters

- **WWDGx:** WWDG Instance

Return values

- **7:** bit Watchdog Counter value

Reference Manual to LL API cross reference:

- CR T LL_WWDG_GetCounter

LL_WWDG_SetPrescaler

Function name

__STATIC_INLINE void LL_WWDG_SetPrescaler (WWDG_TypeDef * WWDGx, uint32_t Prescaler)

Function description

Set the time base of the prescaler (WDGTB).

Parameters

- **WWDGx:** WWDG Instance
- **Prescaler:** This parameter can be one of the following values:
  - LL_WWDG_PRESCALER_1
  - LL_WWDG_PRESCALER_2
  - LL_WWDG_PRESCALER_4
  - LL_WWDG_PRESCALER_8

Return values

- **None:**

Notes

- Prescaler is used to apply ratio on PCLK clock, so that Watchdog counter is decremented every (4096 x 2expWDGTB) PCLK cycles

Reference Manual to LL API cross reference:

- CFR WDGTB LL_WWDG_SetPrescaler

LL_WWDG_GetPrescaler

Function name

__STATIC_INLINE uint32_t LL_WWDG_GetPrescaler (WWDG_TypeDef * WWDGx)

Function description

Return current Watchdog Prescaler Value.

Parameters

- **WWDGx:** WWDG Instance

Return values

- **Returned:** value can be one of the following values:
  - LL_WWDG_PRESCALER_1
  - LL_WWDG_PRESCALER_2
  - LL_WWDG_PRESCALER_4
  - LL_WWDG_PRESCALER_8

Reference Manual to LL API cross reference:

- CFR WDGTB LL_WWDG_GetPrescaler

LL_WWDG_SetWindow

Function name

__STATIC_INLINE void LL_WWDG_SetWindow (WWDG_TypeDef * WWDGx, uint32_t Window)

Function description

Set the Watchdog Window value to be compared to the downcounter (7-bits W[6:0]).

Parameters

- **WWDGx:** WWDG Instance
- **Window:** 0x00..0x7F (7 bit Window value)

Return values

- **None:**

Notes

- This window value defines when write in the WWDG_CR register to program Watchdog counter is allowed. Watchdog counter value update must occur only when the counter value is lower than the Watchdog window register value. Otherwise, a MCU reset is generated if the 7-bit Watchdog counter value (in the control register) is refreshed before the downcounter has reached the watchdog window register value. Physically is possible to set the Window lower then 0x40 but it is not recommended. To generate an immediate reset, it is possible to set the Counter lower than 0x40.

Reference Manual to LL API cross reference:

- CFR W LL_WWDG_SetWindow

LL_WWDG_GetWindow

Function name

__STATIC_INLINE uint32_t LL_WWDG_GetWindow (WWDG_TypeDef * WWDGx)

Function description

Return current Watchdog Window Value (7 bits value)

Parameters

- **WWDGx:** WWDG Instance

Return values

- **7:** bit Watchdog Window value

Reference Manual to LL API cross reference:

- CFR W LL_WWDG_GetWindow

LL_WWDG_IsActiveFlag_EWKUP

Function name

__STATIC_INLINE uint32_t LL_WWDG_IsActiveFlag_EWKUP (WWDG_TypeDef * WWDGx)

Function description

Indicates if the WWDG Early Wakeup Interrupt Flag is set or not.

Parameters

- **WWDGx:** WWDG Instance

Return values

- **State:** of bit (1 or 0).

Notes

- This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. A write of 1 has no effect. This bit is also set if the interrupt is not enabled.

Reference Manual to LL API cross reference:

- SR EWIF LL_WWDG_IsActiveFlag_EWKUP

LL_WWDG_ClearFlag_EWKUP

Function name

__STATIC_INLINE void LL_WWDG_ClearFlag_EWKUP (WWDG_TypeDef * WWDGx)

Function description

Clear WWDG Early Wakeup Interrupt Flag (EWIF)

Parameters

- **WWDGx:** WWDG Instance

Return values

- **None:**

Reference Manual to LL API cross reference:

- SR EWIF LL_WWDG_ClearFlag_EWKUP

LL_WWDG_EnableIT_EWKUP

Function name

__STATIC_INLINE void LL_WWDG_EnableIT_EWKUP (WWDG_TypeDef * WWDGx)

Function description

Enable the Early Wakeup Interrupt.

Parameters

- **WWDGx:** WWDG Instance

Return values

- **None:**

Notes

- When set, an interrupt occurs whenever the counter reaches value 0x40. This interrupt is only cleared by hardware after a reset

Reference Manual to LL API cross reference:

- CFR EWI LL_WWDG_EnableIT_EWKUP

LL_WWDG_IsEnabledIT_EWKUP

Function name

__STATIC_INLINE uint32_t LL_WWDG_IsEnabledIT_EWKUP (WWDG_TypeDef * WWDGx)

Function description

Check if Early Wakeup Interrupt is enabled.

Parameters

- **WWDGx:** WWDG Instance

Return values

- **State:** of bit (1 or 0).

Reference Manual to LL API cross reference:

- CFR EWI LL_WWDG_IsEnabledIT_EWKUP

95.2 WWDG Firmware driver defines

The following section lists the various define and macros of the module.

95.2.1 WWDG

WWDG

IT Defines

LL_WWDG_CFR_EWI

PRESCALER

LL_WWDG_PRESCALER_1

WWDG counter clock = (PCLK1/4096)/1

LL_WWDG_PRESCALER_2

WWDG counter clock = (PCLK1/4096)/2

LL_WWDG_PRESCALER_4

WWDG counter clock = (PCLK1/4096)/4

LL_WWDG_PRESCALER_8

WWDG counter clock = (PCLK1/4096)/8

Common Write and read registers macros

LL_WWDG_WriteReg

Description:

- Write a value in WWDG register.

Parameters:

- __INSTANCE__: WWDG Instance
- __REG__: Register to be written
- __VALUE__: Value to be written in the register

Return value:

- None

LL_WWDG_ReadReg

Description:

- Read a value in WWDG register.

Parameters:

- __INSTANCE__: WWDG Instance
- __REG__: Register to be read

Return value:

- Register: value

96 FAQs

General subjects

Why should I use the HAL drivers?

There are many advantages in using the HAL drivers:

- Ease of use: you can use the HAL drivers to configure and control any peripheral embedded within your STM32 MCU without prior in-depth knowledge of the product.
- HAL drivers provide intuitive and ready-to-use APIs to configure the peripherals and support polling, interrupt and DMA programming model to accommodate all application requirements, thus allowing the end-user to build a complete application by calling a few APIs.
- Higher level of abstraction than a standard peripheral library allowing to transparently manage:
  - Data transfers and processing using blocking mode (polling) or non-blocking mode (interrupt or DMA)
  - Error management through peripheral error detection and timeout mechanism.
- Generic architecture speeding up initialization and porting, thus allowing customers to focus on innovation.
- Generic set of APIs with full compatibility across the STM32 Series/lines, to ease the porting task between STM32 MCUs.
- The APIs provided within the HAL drivers are feature-oriented and do not require in-depth knowledge of peripheral operation.
- The APIs provided are modular. They include initialization, IO operation and control functions. The end-user has to call init function, then start the process by calling one IO operation functions (write, read, transmit, receive, …). Most of the peripherals have the same architecture.
- The number of functions required to build a complete and useful application is very reduced. As an example, to build a UART communication process, the user only has to call HAL_UART_Init() then HAL_UART_Transmit() or HAL_UART_Receive().

Which devices are supported by the HAL drivers?

The HAL drivers are developed to support all STM32F4 devices. To ensure compatibility between all devices and portability with others Series and lines, the API is split into the generic and the extension APIs . For more details, please refer to *section* ***Devices supported by the HAL drivers***.

What is the cost of using HAL drivers in term of code size and performance?

Like generic architecture drivers, the HAL drivers may induce firmware overhead.

This is due to the high abstraction level and ready-to-use APIs which allow data transfers, errors management and offloads the user application from implementation details.

Architecture

How many files should I modify to configure the HAL drivers?

Only one file needs to be modified: stm32f4xx_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 (stm32f4xx_hal_conf_template.c).

Which header files should I include in my application to use the HAL drivers?

Only stm32f4xx_hal.h file has to be included.

What is the difference between xx_hal_ppp.c/.h and xx_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 (stm32f4xx_hal_ppp.c): It includes the common set of APIs across all the STM32 product lines
- The extension APIs (stm32f4xx_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_stm32f4xx.c) but in the main user application by calling the two main functions, HAL_RCC_OscConfig() and HAL_RCC_ClockConfig(). It can be modified in any user application section.

What is the purpose of the *PPP_HandleTypeDef *pHandle* structure located in each driver in addition to the Initialization structure

PPP_HandleTypeDef *pHandle is the main structure implemented in the HAL drivers. It handles the peripheral configuration and registers, and embeds all the structures and variables required to follow the peripheral device flow (pointer to buffer, Error code, State,...)

However, this structure is not required to service peripherals such as GPIO, SYSTICK, PWR, and RCC.

What is the purpose of HAL_PPP_MspInit() and HAL_PPP_MspDeInit() functions?

These function are called within HAL_PPP_Init() and HAL_PPP_DeInit(), respectively. They are used to perform the low level Initialization/de-initialization related to the additional hardware resources (RCC, GPIO, NVIC and DMA).

These functions are declared in stm32f4xx_hal_msp.c. A template is provided in the HAL driver folders (stm32f4xx_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() instm32f4xx_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 stm32f4xx_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 xx_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.

When should I use HAL versus LL drivers?

HAL drivers offer high-level and function-oriented APIs, with a high level of portability. Product/IPs complexity is hidden for end users. LL drivers offer low-level APIs at registers level, with a better optimization but less portability. They require a deep knowledge of product/IPs specifications.

How can I include LL drivers in my environment? Is there any LL configuration file as for HAL?

There is no configuration file. Source code shall directly include the necessary stm32f4xx_ll_ppp.h file(s).

Can I use HAL and LL drivers together? If yes, what are the constraints?

It is possible to use both HAL and LL drivers. One can handle the IP initialization phase with HAL and then manage the I/O operations with LL drivers. The major difference between HAL and LL is that HAL drivers require to create and use handles for operation management while LL drivers operates directly on peripheral registers. Mixing HAL and LL is illustrated in Examples_MIX example.

Is there any LL APIs which are not available with HAL?

Yes, there are. A few Cortex® APIs have been added in stm32f4xx_ll_cortex.h e.g. for accessing SCB or SysTick registers.

Why are SysTick interrupts not enabled on LL drivers?

When using LL drivers in standalone mode, you do not need to enable SysTick interrupts because they are not used in LL APIs, while HAL functions requires SysTick interrupts to manage timeouts.

Revision history

Table 25. **Document revision history**

Date	Revision	Changes
05-May-2014	1	Initial release.
03-Apr-2015	2	Added CEC, FMPI2C, QSPI and SPDIFRX in Table 1. Added STM32F446xx, cec, dcmi, fmpi2c, fmpi2c_ex, spdifrx and qspi in Table 1. Updated Common macros section in HAL common resources. Added HAL CEC Generic Driver, HAL FLASH__RAMFUNC Generic Driver HAL FMPI2C Generic Driver, HAL FMPI2C Extension Driver, HAL QSPI Generic Driver and HAL SPDIFRX Generic Driver.
15-Sep-2015	3	Added DSI and LPTIM and removed msp_template in Table 1. Added STM32F469xx, STM32F479xx, STM32F410xx, dsi, ltdc_ex and lptim in Table 1. Added HAL DSI Generic Driver, HAL LPTIM Generic Driver and HAL LTDC Generic Driver.
02-Sep-2016	4	Added DFSDM in Acronyms and definitions. Added STM32F412Cx, STM32F412Rx, STM32F412Vx, STM32F412Zx and DFSDM in Table 1. Added HAL DFSDM Generic Driver.
20-Feb-2017	5	Added MMC in Section 2 Acronyms and definitions. Added STM32F413xx, STM32F423xx, MMC row and LL driver rows in Table : "List of devices supported by HAL drivers". Added Section 43: "HAL MMC Generic Driver" description. Added description of LL Generic drivers. Updated FAQ section.
31-Jul-2020	6	Updated Section Introduction. Added Section 1 General information. Updated Section 2 Acronyms and definitions. Updated Figure 1. Example of project template. Added Section 4 Overview of low-layer drivers and Section 5 Cohabiting of HAL and LL. Updated HAL drivers. Added low-layer drivers.
18-Jun-2021	7	Section 40 HAL IRDA Generic Driver: updated note at the end of Section 40.2.2 Callback registration. Section 70 HAL UART Generic Driver: • Added support for the following new APIs: HAL_UARTEx_ReceiveToIdle(), HAL_UARTEx_ReceiveToIdle_IT() and HAL_UARTEx_ReceiveToIdle_DMA(). • Updated note at the end of Section 70.2.2 Callback registration. Section 71 HAL USART Generic Driver: updated note at the end of Section 71.2.2 Callback registration.

Contents

List of tables

List of figures