20 HAL DMA Generic Driver

20.1 DMA Firmware driver registers structures

20.1.1 DMA_InitTypeDef

Data Fields

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

Field Documentation

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

20.1.2 __DMA_HandleTypeDef

Data Fields

- ***DMA_Channel_TypeDef * Instance***
- ***DMA_InitTypeDef Init***
- ***HAL_LockTypeDef Lock***

- ***__IO HAL_DMA_StateTypeDef State***
- ***void * Parent***
- ***void(* XferCpltCallback***
- ***void(* XferHalfCpltCallback***
- ***void(* XferErrorCallback***
- ***void(* XferAbortCallback***
- ***__IO uint32_t ErrorCode***
- ***DMA_TypeDef * DmaBaseAddress***
- ***uint32_t ChannelIndex***
- ***DMAMUX_Channel_TypeDef * DMAmuxChannel***
- ***DMAMUX_ChannelStatus_TypeDef * DMAmuxChannelStatus***
- ***uint32_t DMAmuxChannelStatusMask***
- ***DMAMUX_RequestGen_TypeDef * DMAmuxRequestGen***
- ***DMAMUX_RequestGenStatus_TypeDef * DMAmuxRequestGenStatus***
- ***uint32_t DMAmuxRequestGenStatusMask***

Field Documentation

- ***DMA_Channel_TypeDef* __DMA_HandleTypeDef::Instance***
  Register base address
- ***DMA_InitTypeDef __DMA_HandleTypeDef::Init***
  DMA communication parameters
- ***HAL_LockTypeDef __DMA_HandleTypeDef::Lock***
  DMA locking object
- ***__IO HAL_DMA_StateTypeDef __DMA_HandleTypeDef::State***
  DMA transfer state
- ***void* __DMA_HandleTypeDef::Parent***
  Parent object state
- ***void(* __DMA_HandleTypeDef::XferCpltCallback)(struct __DMA_HandleTypeDef *hdma)***
  DMA transfer complete callback
- ***void(* __DMA_HandleTypeDef::XferHalfCpltCallback)(struct __DMA_HandleTypeDef *hdma)***
  DMA Half transfer complete callback
- ***void(* __DMA_HandleTypeDef::XferErrorCallback)(struct __DMA_HandleTypeDef *hdma)***
  DMA transfer error callback
- ***void(* __DMA_HandleTypeDef::XferAbortCallback)(struct __DMA_HandleTypeDef *hdma)***
  DMA transfer abort callback
- ***__IO uint32_t __DMA_HandleTypeDef::ErrorCode***
  DMA Error code
- ***DMA_TypeDef* __DMA_HandleTypeDef::DmaBaseAddress***
  DMA Channel Base Address
- ***uint32_t __DMA_HandleTypeDef::ChannelIndex***
  DMA Channel Index
- ***DMAMUX_Channel_TypeDef* __DMA_HandleTypeDef::DMAmuxChannel***
  Register base address
- ***DMAMUX_ChannelStatus_TypeDef* __DMA_HandleTypeDef::DMAmuxChannelStatus***
  DMAMUX Channels Status Base Address
- ***uint32_t __DMA_HandleTypeDef::DMAmuxChannelStatusMask***
  DMAMUX Channel Status Mask
- ***DMAMUX_RequestGen_TypeDef* __DMA_HandleTypeDef::DMAmuxRequestGen***
  DMAMUX request generator Base Address

- ***DMAMUX_RequestGenStatus_TypeDef* __DMA_HandleTypeDef::DMAmuxRequestGenStatus***
  DMAMUX request generator Address
- ***uint32_t __DMA_HandleTypeDef::DMAmuxRequestGenStatusMask***
  DMAMUX request generator Status mask

20.2 DMA Firmware driver API description

20.2.1 How to use this driver

1. Enable and configure the peripheral to be connected to the DMA Channel (except for internal SRAM / FLASH memories: no initialization is necessary). Please refer to the Reference manual for connection between peripherals and DMA requests.
2. For a given Channel, program the required configuration through the following parameters: Channel request, Transfer Direction, Source and Destination data formats, Circular or Normal mode, Channel Priority level, Source and Destination Increment mode using HAL_DMA_Init() function. Prior to HAL_DMA_Init the peripheral clock shall be enabled for both DMA & DMAMUX thanks to:
   a. DMA1 or DMA2: __HAL_RCC_DMA1_CLK_ENABLE() or __HAL_RCC_DMA2_CLK_ENABLE() ;
   b. DMAMUX1: __HAL_RCC_DMAMUX1_CLK_ENABLE();
3. Use HAL_DMA_GetState() function to return the DMA state and HAL_DMA_GetError() in case of error detection.
4. Use HAL_DMA_Abort() function to abort the current transfer In Memory-to-Memory transfer mode, Circular mode is not allowed.

Polling mode IO operation

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

Interrupt mode IO operation

- Configure the DMA interrupt priority using HAL_NVIC_SetPriority()
- Enable the DMA IRQ handler using HAL_NVIC_EnableIRQ()
- Use HAL_DMA_Start_IT() to start DMA transfer after the configuration of Source address and destination address and the Length of data to be transferred. In this case the DMA interrupt is configured
- Use HAL_DMA_IRQHandler() called under DMA_IRQHandler() Interrupt subroutine
- At the end of data transfer HAL_DMA_IRQHandler() function is executed and user can add his own function to register callbacks with HAL_DMA_RegisterCallback().

DMA HAL driver macros list

Below the list of macros in DMA HAL driver.

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

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

20.2.2 Initialization and de-initialization functions

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

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

This section contains the following APIs:

- ***HAL_DMA_Init()***
- ***HAL_DMA_DeInit()***

20.2.3 IO operation functions

This section provides functions allowing to:

- Configure the source, destination address and data length and Start DMA transfer
- Configure the source, destination address and data length and Start DMA transfer with interrupt
- Abort DMA transfer
- Poll for transfer complete
- Handle DMA interrupt request

This section contains the following APIs:

- ***HAL_DMA_Start()***
- ***HAL_DMA_Start_IT()***
- ***HAL_DMA_Abort()***
- ***HAL_DMA_Abort_IT()***
- ***HAL_DMA_PollForTransfer()***
- ***HAL_DMA_IRQHandler()***
- ***HAL_DMA_RegisterCallback()***
- ***HAL_DMA_UnRegisterCallback()***

20.2.4 Peripheral State and Errors functions

This subsection provides functions allowing to

- Check the DMA state
- Get error code

This section contains the following APIs:

- ***HAL_DMA_GetState()***
- ***HAL_DMA_GetError()***

20.2.5 Detailed description of functions

HAL_DMA_Init

Function name	**HAL_StatusTypeDef HAL_DMA_Init (DMA_HandleTypeDef * hdma)**

Function description: Initialize the DMA according to the specified parameters in the DMA_InitTypeDef and initialize the associated handle.

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

Return values:
- **HAL:** status

HAL_DMA_DeInit

Function name: **HAL_StatusTypeDef HAL_DMA_DeInit (DMA_HandleTypeDef * hdma)**

Function description: DeInitialize the DMA peripheral.

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

Return values:
- **HAL:** status

HAL_DMA_Start

Function name: **HAL_StatusTypeDef HAL_DMA_Start (DMA_HandleTypeDef * hdma, uint32_t SrcAddress, uint32_t DstAddress, uint32_t DataLength)**

Function description: Start the DMA Transfer.

Parameters:
- **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
- **SrcAddress:** The source memory Buffer address
- **DstAddress:** The destination memory Buffer address
- **DataLength:** The length of data to be transferred from source to destination

Return values:
- **HAL:** status

HAL_DMA_Start_IT

Function name: **HAL_StatusTypeDef HAL_DMA_Start_IT (DMA_HandleTypeDef * hdma, uint32_t SrcAddress, uint32_t DstAddress, uint32_t DataLength)**

Function description: Start the DMA Transfer with interrupt enabled.

Parameters:
- **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
- **SrcAddress:** The source memory Buffer address
- **DstAddress:** The destination memory Buffer address
- **DataLength:** The length of data to be transferred from source to destination

Return values:
- **HAL:** status

HAL_DMA_Abort

Function name **HAL_StatusTypeDef HAL_DMA_Abort (DMA_HandleTypeDef * hdma)**

Function description Abort the DMA Transfer.

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

Return values
- **HAL:** status

HAL_DMA_Abort_IT

Function name **HAL_StatusTypeDef HAL_DMA_Abort_IT (DMA_HandleTypeDef * hdma)**

Function description Aborts the DMA Transfer in Interrupt mode.

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

Return values
- **HAL:** status

HAL_DMA_PollForTransfer

Function name **HAL_StatusTypeDef HAL_DMA_PollForTransfer (DMA_HandleTypeDef * hdma, HAL_DMA_LevelCompleteTypeDef CompleteLevel, uint32_t Timeout)**

Function description Polling for transfer complete.

Parameters
- **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
- **CompleteLevel:** Specifies the DMA level complete.
- **Timeout:** Timeout duration.

Return values
- **HAL:** status

HAL_DMA_IRQHandler

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

Function description Handle DMA interrupt request.

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

Return values
- **None:**

HAL_DMA_RegisterCallback

Function name **HAL_StatusTypeDef HAL_DMA_RegisterCallback (DMA_HandleTypeDef * hdma, HAL_DMA_CallbackIDTypeDef CallbackID, void(*)(DMA_HandleTypeDef *_hdma) pCallback)**

Function description	Register callbacks.
Parameters	• **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel. • **CallbackID:** User Callback identifer a HAL_DMA_CallbackIDTypeDef ENUM as parameter. • **pCallback:** pointer to private callbacsk function which has pointer to a DMA_HandleTypeDef structure as parameter.
Return values	• **HAL:** status

HAL_DMA_UnRegisterCallback

Function name	**HAL_StatusTypeDef HAL_DMA_UnRegisterCallback (DMA_HandleTypeDef * hdma, HAL_DMA_CallbackIDTypeDef CallbackID)**
Function description	UnRegister callbacks.
Parameters	• **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel. • **CallbackID:** User Callback identifer a HAL_DMA_CallbackIDTypeDef ENUM as parameter.
Return values	• **HAL:** status

HAL_DMA_GetState

Function name	**HAL_DMA_StateTypeDef HAL_DMA_GetState (DMA_HandleTypeDef * hdma)**
Function description	Return the DMA hande state.
Parameters	• **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
Return values	• **HAL:** state

HAL_DMA_GetError

Function name	**uint32_t HAL_DMA_GetError (DMA_HandleTypeDef * hdma)**
Function description	Return the DMA error code.
Parameters	• **hdma:** : pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA Channel.
Return values	• **DMA:** Error Code

20.3 DMA Firmware driver defines

20.3.1 DMA

DMA Data transfer direction

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

DMA Error Code

HAL_DMA_ERROR_NONE	No error
HAL_DMA_ERROR_TE	Transfer error
HAL_DMA_ERROR_NO_XFER	Abort requested with no Xfer ongoing
HAL_DMA_ERROR_TIMEOUT	Timeout error
HAL_DMA_ERROR_NOT_SUPPORTED	Not supported mode
HAL_DMA_ERROR_SYNC	DMAMUX sync overrun error
HAL_DMA_ERROR_REQGEN	DMAMUX request generator overrun error

DMA Exported Macros

__HAL_DMA_RESET_HANDLE_STATE

Description:

- Reset DMA handle state.

Parameters:

- __HANDLE__: DMA handle

Return value:

- None

__HAL_DMA_ENABLE

Description:

- Enable the specified DMA Channel.

Parameters:

- __HANDLE__: DMA handle

Return value:

- None

__HAL_DMA_DISABLE

Description:

- Disable the specified DMA Channel.

Parameters:

- __HANDLE__: DMA handle

Return value:

- None

__HAL_DMA_GET_TC_FLAG_INDEX

Description:

- Return the current DMA Channel transfer complete flag.

Parameters:

- __HANDLE__: DMA handle

Return value:

- The: specified transfer complete flag

index.

__HAL_DMA_GET_HT_FLAG_INDEX

Description:

- Return the current DMA Channel half transfer complete flag.

Parameters:

- __HANDLE__: DMA handle

Return value:

- The: specified half transfer complete flag index.

__HAL_DMA_GET_TE_FLAG_INDEX

Description:

- Return the current DMA Channel transfer error flag.

Parameters:

- __HANDLE__: DMA handle

Return value:

- The: specified transfer error flag index.

__HAL_DMA_GET_GI_FLAG_INDEX

Description:

- Return the current DMA Channel Global interrupt flag.

Parameters:

- __HANDLE__: DMA handle

Return value:

- The: specified transfer error flag index.

__HAL_DMA_GET_FLAG

Description:

- Get the DMA Channel pending flags.

Parameters:

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

Return value:

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

__HAL_DMA_CLEAR_FLAG

Description:

- Clear the DMA Channel pending flags.

Parameters:

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

Return value:

- None

__HAL_DMA_ENABLE_IT

Description:

- Enable the specified DMA Channel interrupts.

Parameters:

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

Return value:

- None

__HAL_DMA_DISABLE_IT

Description:

- Disable the specified DMA Channel interrupts.

Parameters:

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

mask

Return value:

- None

__HAL_DMA_GET_IT_SOURCE

Description:

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

Parameters:

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

Return value:

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

__HAL_DMA_GET_COUNTER

Description:

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

Parameters:

- __HANDLE__: DMA handle

Return value:

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

DMA flag definitions

DMA_FLAG_GL1

DMA_FLAG_TC1

DMA_FLAG_HT1

DMA_FLAG_TE1

DMA_FLAG_GL2

DMA_FLAG_TC2

DMA_FLAG_HT2

DMA_FLAG_TE2

DMA_FLAG_GL3

DMA_FLAG_TC3

DMA_FLAG_HT3

DMA_FLAG_TE3

DMA_FLAG_GL4

DMA_FLAG_TC4

DMA_FLAG_HT4

DMA_FLAG_TE4

DMA_FLAG_GL5

DMA_FLAG_TC5

DMA_FLAG_HT5

DMA_FLAG_TE5

DMA_FLAG_GL6

DMA_FLAG_TC6

DMA_FLAG_HT6

DMA_FLAG_TE6

DMA_FLAG_GL7

DMA_FLAG_TC7

DMA_FLAG_HT7

DMA_FLAG_TE7

TIM DMA Handle Index

TIM_DMA_ID_UPDATE	Index of the DMA handle used for Update DMA requests
TIM_DMA_ID_CC1	Index of the DMA handle used for Capture/Compare 1 DMA requests
TIM_DMA_ID_CC2	Index of the DMA handle used for Capture/Compare 2 DMA requests
TIM_DMA_ID_CC3	Index of the DMA handle used for Capture/Compare 3 DMA requests
TIM_DMA_ID_CC4	Index of the DMA handle used for Capture/Compare 4 DMA requests
TIM_DMA_ID_COMMUTATION	Index of the DMA handle used for Commutation DMA requests
TIM_DMA_ID_TRIGGER	Index of the DMA handle used for Trigger DMA requests

DMA interrupt enable definitions

DMA_IT_TC

DMA_IT_HT

DMA_IT_TE

DMA Memory data size

DMA_MDATAALIGN_BYTE	Memory data alignment: Byte
DMA_MDATAALIGN_HALFWORD	Memory data alignment: HalfWord
DMA_MDATAALIGN_WORD	Memory data alignment: Word

DMA Memory incremented mode

DMA_MINC_ENABLE	Memory increment mode Enable
DMA_MINC_DISABLE	Memory increment mode Disable

DMA mode

DMA_NORMAL	Normal mode
DMA_CIRCULAR	Circular mode

DMA Peripheral data size

DMA_PDATAALIGN_BYTE	Peripheral data alignment: Byte
DMA_PDATAALIGN_HALFWORD	Peripheral data alignment: HalfWord
DMA_PDATAALIGN_WORD	Peripheral data alignment: Word

DMA Peripheral incremented mode

DMA_PINC_ENABLE	Peripheral increment mode Enable
DMA_PINC_DISABLE	Peripheral increment mode Disable

DMA Priority level

DMA_PRIORITY_LOW	Priority level: Low
DMA_PRIORITY_MEDIUM	Priority level: Medium
DMA_PRIORITY_HIGH	Priority level: High
DMA_PRIORITY_VERY_HIGH	Priority level: Very_High

DMA request

DMA_REQUEST_MEM2MEM	memory to memory transfer
DMA_REQUEST_GENERATOR0	DMAMUX1 request generator 0
DMA_REQUEST_GENERATOR1	DMAMUX1 request generator 1
DMA_REQUEST_GENERATOR2	DMAMUX1 request generator 2
DMA_REQUEST_GENERATOR3	DMAMUX1 request generator 3
DMA_REQUEST_ADC1	DMAMUX1 ADC1 request
DMA_REQUEST_DAC1_CH1	DMAMUX1 DAC1 CH1 request
DMA_REQUEST_DAC1_CH2	DMAMUX1 DAC1 CH2 request
DMA_REQUEST_TIM6_UP	DMAMUX1 TIM6 UP request
DMA_REQUEST_TIM7_UP	DMAMUX1 TIM7 UP request
DMA_REQUEST_SPI1_RX	DMAMUX1 SPI1 RX request
DMA_REQUEST_SPI1_TX	DMAMUX1 SPI1 TX request
DMA_REQUEST_SPI2_RX	DMAMUX1 SPI2 RX request
DMA_REQUEST_SPI2_TX	DMAMUX1 SPI2 TX request
DMA_REQUEST_SPI3_RX	DMAMUX1 SPI3 RX request
DMA_REQUEST_SPI3_TX	DMAMUX1 SPI3 TX request
DMA_REQUEST_I2C1_RX	DMAMUX1 I2C1 RX request

DMA_REQUEST_I2C1_TX	DMAMUX1 I2C1 TX request
DMA_REQUEST_I2C2_RX	DMAMUX1 I2C2 RX request
DMA_REQUEST_I2C2_TX	DMAMUX1 I2C2 TX request
DMA_REQUEST_I2C3_RX	DMAMUX1 I2C3 RX request
DMA_REQUEST_I2C3_TX	DMAMUX1 I2C3 TX request
DMA_REQUEST_I2C4_RX	DMAMUX1 I2C4 RX request
DMA_REQUEST_I2C4_TX	DMAMUX1 I2C4 TX request
DMA_REQUEST_USART1_RX	DMAMUX1 USART1 RX request
DMA_REQUEST_USART1_TX	DMAMUX1 USART1 TX request
DMA_REQUEST_USART2_RX	DMAMUX1 USART2 RX request
DMA_REQUEST_USART2_TX	DMAMUX1 USART2 TX request
DMA_REQUEST_USART3_RX	DMAMUX1 USART3 RX request
DMA_REQUEST_USART3_TX	DMAMUX1 USART3 TX request
DMA_REQUEST_UART4_RX	DMAMUX1 UART4 RX request
DMA_REQUEST_UART4_TX	DMAMUX1 UART4 TX request
DMA_REQUEST_UART5_RX	DMAMUX1 UART5 RX request
DMA_REQUEST_UART5_TX	DMAMUX1 UART5 TX request
DMA_REQUEST_LPUART1_RX	DMAMUX1 LP_UART1_RX request
DMA_REQUEST_LPUART1_TX	DMAMUX1 LP_UART1_RX request
DMA_REQUEST_SAI1_A	DMAMUX1 SAI1 A request
DMA_REQUEST_SAI1_B	DMAMUX1 SAI1 B request
DMA_REQUEST_SAI2_A	DMAMUX1 SAI2 A request
DMA_REQUEST_SAI2_B	DMAMUX1 SAI2 B request
DMA_REQUEST_OCTOSPI1	DMAMUX1 OCTOSPI1 request
DMA_REQUEST_OCTOSPI2	DMAMUX1 OCTOSPI2 request
DMA_REQUEST_TIM1_CH1	DMAMUX1 TIM1 CH1 request
DMA_REQUEST_TIM1_CH2	DMAMUX1 TIM1 CH2 request
DMA_REQUEST_TIM1_CH3	DMAMUX1 TIM1 CH3 request
DMA_REQUEST_TIM1_CH4	DMAMUX1 TIM1 CH4 request
DMA_REQUEST_TIM1_UP	DMAMUX1 TIM1 UP request
DMA_REQUEST_TIM1_TRIG	DMAMUX1 TIM1 TRIG request
DMA_REQUEST_TIM1_COM	DMAMUX1 TIM1 COM request
DMA_REQUEST_TIM8_CH1	DMAMUX1 TIM8 CH1 request
DMA_REQUEST_TIM8_CH2	DMAMUX1 TIM8 CH2 request
DMA_REQUEST_TIM8_CH3	DMAMUX1 TIM8 CH3 request
DMA_REQUEST_TIM8_CH4	DMAMUX1 TIM8 CH4 request

DMA_REQUEST_TIM8_UP	DMAMUX1 TIM8 UP request
DMA_REQUEST_TIM8_TRIG	DMAMUX1 TIM8 TRIG request
DMA_REQUEST_TIM8_COM	DMAMUX1 TIM8 COM request
DMA_REQUEST_TIM2_CH1	DMAMUX1 TIM2 CH1 request
DMA_REQUEST_TIM2_CH2	DMAMUX1 TIM2 CH2 request
DMA_REQUEST_TIM2_CH3	DMAMUX1 TIM2 CH3 request
DMA_REQUEST_TIM2_CH4	DMAMUX1 TIM2 CH4 request
DMA_REQUEST_TIM2_UP	DMAMUX1 TIM2 UP request
DMA_REQUEST_TIM3_CH1	DMAMUX1 TIM3 CH1 request
DMA_REQUEST_TIM3_CH2	DMAMUX1 TIM3 CH2 request
DMA_REQUEST_TIM3_CH3	DMAMUX1 TIM3 CH3 request
DMA_REQUEST_TIM3_CH4	DMAMUX1 TIM3 CH4 request
DMA_REQUEST_TIM3_UP	DMAMUX1 TIM3 UP request
DMA_REQUEST_TIM3_TRIG	DMAMUX1 TIM3 TRIG request
DMA_REQUEST_TIM4_CH1	DMAMUX1 TIM4 CH1 request
DMA_REQUEST_TIM4_CH2	DMAMUX1 TIM4 CH2 request
DMA_REQUEST_TIM4_CH3	DMAMUX1 TIM4 CH3 request
DMA_REQUEST_TIM4_CH4	DMAMUX1 TIM4 CH4 request
DMA_REQUEST_TIM4_UP	DMAMUX1 TIM4 UP request
DMA_REQUEST_TIM5_CH1	DMAMUX1 TIM5 CH1 request
DMA_REQUEST_TIM5_CH2	DMAMUX1 TIM5 CH2 request
DMA_REQUEST_TIM5_CH3	DMAMUX1 TIM5 CH3 request
DMA_REQUEST_TIM5_CH4	DMAMUX1 TIM5 CH4 request
DMA_REQUEST_TIM5_UP	DMAMUX1 TIM5 UP request
DMA_REQUEST_TIM5_TRIG	DMAMUX1 TIM5 TRIG request
DMA_REQUEST_TIM15_CH1	DMAMUX1 TIM15 CH1 request
DMA_REQUEST_TIM15_UP	DMAMUX1 TIM15 UP request
DMA_REQUEST_TIM15_TRIG	DMAMUX1 TIM15 TRIG request
DMA_REQUEST_TIM15_COM	DMAMUX1 TIM15 COM request
DMA_REQUEST_TIM16_CH1	DMAMUX1 TIM16 CH1 request
DMA_REQUEST_TIM16_UP	DMAMUX1 TIM16 UP request
DMA_REQUEST_TIM17_CH1	DMAMUX1 TIM17 CH1 request
DMA_REQUEST_TIM17_UP	DMAMUX1 TIM17 UP request
DMA_REQUEST_DFSDM1_FLT0	DMAMUX1 DFSDM1 Filter0 request
DMA_REQUEST_DFSDM1_FLT1	DMAMUX1 DFSDM1 Filter1 request
DMA_REQUEST_DFSDM1_FLT2	DMAMUX1 DFSDM1 Filter2 request

DMA_REQUEST_DFSDM1_FLT3	DMAMUX1 DFSDM1 Filter3 request
DMA_REQUEST_DCMI	DMAMUX1 DCMI request
DMA_REQUEST_AES_IN	DMAMUX1 AES IN request
DMA_REQUEST_AES_OUT	DMAMUX1 AES OUT request
DMA_REQUEST_HASH_IN	DMAMUX1 HASH IN request

21 HAL DMA Extension Driver

21.1 DMAEx Firmware driver registers structures

21.1.1 HAL_DMA_MuxSyncConfigTypeDef

Data Fields

- ***uint32_t SyncSignalID***
- ***uint32_t SyncPolarity***
- ***FunctionalState SyncEnable***
- ***FunctionalState EventEnable***
- ***uint32_t RequestNumber***

Field Documentation

- ***uint32_t HAL_DMA_MuxSyncConfigTypeDef::SyncSignalID***
  Specifies the synchronization signal gating the DMA request in periodic mode. This parameter can be a value of ***DMAEx_DMAMUX_SyncSignalID_selection***
- ***uint32_t HAL_DMA_MuxSyncConfigTypeDef::SyncPolarity***
  Specifies the polarity of the signal on which the DMA request is synchronized. This parameter can be a value of ***DMAEx_DMAMUX_SyncPolarity_selection***
- ***FunctionalState HAL_DMA_MuxSyncConfigTypeDef::SyncEnable***
  Specifies if the synchronization shall be enabled or disabled This parameter can take the value ENABLE or DISABLE
- ***FunctionalState HAL_DMA_MuxSyncConfigTypeDef::EventEnable***
  Specifies if an event shall be generated once the RequestNumber is reached. This parameter can take the value ENABLE or DISABLE
- ***uint32_t HAL_DMA_MuxSyncConfigTypeDef::RequestNumber***
  Specifies the number of DMA request that will be authorized after a sync event This parameter must be a number between Min_Data = 1 and Max_Data = 32

21.1.2 HAL_DMA_MuxRequestGeneratorConfigTypeDef

Data Fields

- ***uint32_t SignalID***
- ***uint32_t Polarity***
- ***uint32_t RequestNumber***

Field Documentation

- ***uint32_t HAL_DMA_MuxRequestGeneratorConfigTypeDef::SignalID***
  Specifies the ID of the signal used for DMAMUX request generator This parameter can be a value of ***DMAEx_DMAMUX_SignalGeneratorID_selection***
- ***uint32_t HAL_DMA_MuxRequestGeneratorConfigTypeDef::Polarity***
  Specifies the polarity of the signal on which the request is generated. This parameter can be a value of ***DMAEx_DMAMUX_RequestGeneneratorPolarity_selection***
- ***uint32_t HAL_DMA_MuxRequestGeneratorConfigTypeDef::RequestNumber***
  Specifies the number of DMA request that will be generated after a signal event This parameter must be a number between Min_Data = 1 and Max_Data = 32

21.2 DMAEx Firmware driver API description

21.2.1 How to use this driver

The DMA Extension HAL driver can be used as follows:

- Configure the DMA_MUX Synchronization Block using HAL_DMAEx_ConfigMuxSync function.
- Configure the DMA_MUX Request Generator Block using HAL_DMAEx_ConfigMuxRequestGenerator function. Functions HAL_DMAEx_EnableMuxRequestGenerator and HAL_DMAEx_DisableMuxRequestGenerator can then be used to respectively enable/disable the request generator.
- To handle the DMAMUX Interrupts, the function HAL_DMAEx_MUX_IRQHandler should be called from the DMAMUX IRQ handler i.e DMAMUX1_OVR_IRQHandler. As only one interrupt line is available for all DMAMUX channels and request generators , HAL_DMAEx_MUX_IRQHandler should be called with, as parameter, the appropriate DMA handle as many as used DMAs in the user project (exception done if a given DMA is not using the DMAMUX SYNC block neither a request generator)

21.2.2 Extended features functions

This section provides functions allowing to:

- Configure the DMAMUX Synchronization Block using HAL_DMAEx_ConfigMuxSync function.
- Configure the DMAMUX Request Generator Block using HAL_DMAEx_ConfigMuxRequestGenerator function. Functions HAL_DMAEx_EnableMuxRequestGenerator and HAL_DMAEx_DisableMuxRequestGenerator can then be used to respectively enable/disable the request generator.

This section contains the following APIs:

- ***HAL_DMAEx_ConfigMuxSync()***
- ***HAL_DMAEx_ConfigMuxRequestGenerator()***
- ***HAL_DMAEx_EnableMuxRequestGenerator()***
- ***HAL_DMAEx_DisableMuxRequestGenerator()***
- ***HAL_DMAEx_MUX_IRQHandler()***

21.2.3 Detailed description of functions

HAL_DMAEx_ConfigMuxRequestGenerator

Function name
HAL_StatusTypeDef HAL_DMAEx_ConfigMuxRequestGenerator (DMA_HandleTypeDef * hdma, HAL_DMA_MuxRequestGeneratorConfigTypeDef * pRequestGeneratorConfig)

Function description
Configure the DMAMUX request generator block used by the given DMA channel (instance).

Parameters
- **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA channel.
- **pRequestGeneratorConfig:** : pointer to HAL_DMA_MuxRequestGeneratorConfigTypeDef: contains

the request generator parameters.

Return values
- **HAL:** status

HAL_DMAEx_EnableMuxRequestGenerator

Function name
HAL_StatusTypeDef HAL_DMAEx_EnableMuxRequestGenerator (DMA_HandleTypeDef * hdma)

Function description
Enable the DMAMUX request generator block used by the given DMA channel (instance).

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

Return values
- **HAL:** status

HAL_DMAEx_DisableMuxRequestGenerator

Function name
HAL_StatusTypeDef HAL_DMAEx_DisableMuxRequestGenerator (DMA_HandleTypeDef * hdma)

Function description
Disable the DMAMUX request generator block used by the given DMA channel (instance).

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

Return values
- **HAL:** status

HAL_DMAEx_ConfigMuxSync

Function name
HAL_StatusTypeDef HAL_DMAEx_ConfigMuxSync (DMA_HandleTypeDef * hdma, HAL_DMA_MuxSyncConfigTypeDef * pSyncConfig)

Function description
Configure the DMAMUX synchronization parameters for a given DMA channel (instance).

Parameters
- **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA channel.
- **pSyncConfig:** : pointer to HAL_DMA_MuxSyncConfigTypeDef: contains the DMAMUX synchronization parameters

Return values
- **HAL:** status

HAL_DMAEx_MUX_IRQHandler

Function name
void HAL_DMAEx_MUX_IRQHandler (DMA_HandleTypeDef * hdma)

Function description
Handles DMAMUX interrupt request.

Parameters
- **hdma:** pointer to a DMA_HandleTypeDef structure that

contains the configuration information for the specified DMA channel.

Return values
- **None:**

21.3 DMAEx Firmware driver defines

21.3.1 DMAEx

DMAMUX RequestGeneneratorPolarity selection

HAL_DMAMUX_REQUEST_GEN_NO_EVENT	block request generator events
HAL_DMAMUX_REQUEST_GEN_RISING	generate request on rising edge events
HAL_DMAMUX_REQUEST_GEN_FALLING	generate request on falling edge events
HAL_DMAMUX_REQUEST_GEN_RISING_FALLING	generate request on rising and falling edge events

DMAMUX SignalGeneratorID selection

HAL_DMAMUX1_REQUEST_GEN_EXTI0	Request generator Signal is EXTI0 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI1	Request generator Signal is EXTI1 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI2	Request generator Signal is EXTI2 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI3	Request generator Signal is EXTI3 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI4	Request generator Signal is EXTI4 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI5	Request generator Signal is EXTI5 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI6	Request generator Signal is EXTI6 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI7	Request generator Signal is EXTI7 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI8	Request generator Signal is EXTI8 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI9	Request generator Signal is EXTI9 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI10	Request generator Signal is EXTI10 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI11	Request generator Signal is EXTI11 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI12	Request generator Signal is EXTI12 IT
HAL_DMAMUX1_REQUEST_GEN_EXTI13	Request generator Signal is