# 14 HAL DAC Generic Driver

## 14.1 DAC Firmware driver registers structures

### 14.1.1 DAC_HandleTypeDef

**Data Fields**

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

**Field Documentation**

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

### 14.1.2 DAC_SampleAndHoldConfTypeDef

**Data Fields**

- ***uint32_t DAC_SampleTime***
- ***uint32_t DAC_HoldTime***
- ***uint32_t DAC_RefreshTime***

**Field Documentation**

- ***uint32_t DAC_SampleAndHoldConfTypeDef::DAC_SampleTime***
  Specifies the Sample time for the selected channel. This parameter applies when DAC_SampleAndHold is DAC_SAMPLEANDHOLD_ENABLE. This parameter must be a number between Min_Data = 0 and Max_Data = 1023
- ***uint32_t DAC_SampleAndHoldConfTypeDef::DAC_HoldTime***
  Specifies the hold time for the selected channel This parameter applies when DAC_SampleAndHold is DAC_SAMPLEANDHOLD_ENABLE. This parameter must be a number between Min_Data = 0 and Max_Data = 1023
- ***uint32_t DAC_SampleAndHoldConfTypeDef::DAC_RefreshTime***
  Specifies the refresh time for the selected channel This parameter applies when DAC_SampleAndHold is DAC_SAMPLEANDHOLD_ENABLE. This parameter must be a number between Min_Data = 0 and Max_Data = 255

### 14.1.3 DAC_ChannelConfTypeDef

**Data Fields**

- ***uint32_t DAC_HighFrequency***
- ***uint32_t DAC_SampleAndHold***
- ***uint32_t DAC_Trigger***
- ***uint32_t DAC_OutputBuffer***
- ***uint32_t DAC_ConnectOnChipPeripheral***
- ***uint32_t DAC_UserTrimming***
- ***uint32_t DAC_TrimmingValue***
- ***DAC_SampleAndHoldConfTypeDef DAC_SampleAndHoldConfig***

**Field Documentation**

- ***uint32_t DAC_ChannelConfTypeDef::DAC_HighFrequency***
  Specifies the frequency interface mode This parameter can be a value of ***DAC_HighFrequency***
- ***uint32_t DAC_ChannelConfTypeDef::DAC_SampleAndHold***
  Specifies whether the DAC mode. This parameter can be a value of ***DAC_SampleAndHold***
- ***uint32_t DAC_ChannelConfTypeDef::DAC_Trigger***
  Specifies the external trigger for the selected DAC channel. This parameter can be a value of ***DAC_trigger_selection***
- ***uint32_t DAC_ChannelConfTypeDef::DAC_OutputBuffer***
  Specifies whether the DAC channel output buffer is enabled or disabled. This parameter can be a value of ***DAC_output_buffer***
- ***uint32_t DAC_ChannelConfTypeDef::DAC_ConnectOnChipPeripheral***
  Specifies whether the DAC output is connected or not to on chip peripheral . This parameter can be a value of ***DAC_ConnectOnChipPeripheral***
- ***uint32_t DAC_ChannelConfTypeDef::DAC_UserTrimming***
  Specifies the trimming mode This parameter must be a value of ***DAC_UserTrimming***
  DAC_UserTrimming is either factory or user trimming
- ***uint32_t DAC_ChannelConfTypeDef::DAC_TrimmingValue***
  Specifies the offset trimming value i.e. when DAC_SampleAndHold is DAC_TRIMMING_USER. This parameter must be a number between Min_Data = 1 and Max_Data = 31
- ***DAC_SampleAndHoldConfTypeDef DAC_ChannelConfTypeDef::DAC_SampleAndHoldConfig***
  Sample and Hold settings

## 14.2 DAC Firmware driver API description

### 14.2.1 DAC Peripheral features

#### DAC Channels

STM32L4 devices integrate one or two 12-bit Digital Analog Converters (i.e. one or 2 channel(s)) 1 channel: STM32L451xx STM32L452xx STM32L462xx 2 channels: STM32L431xx STM32L432xx STM32L433xx STM32L442xx STM32L443xx STM32L471xx STM32L475xx STM32L476xx STM32L485xx STM32L486xx STM32L496xx STM32L4A6xx STM32L4R5xx STM32L4R7xx STM32L4R9xx STM32L4S5xx STM32L4S7xx STM32L4S9xx When 2 channels are available, the 2 converters (i.e. channel1 & channel2) can be used independently or simultaneously (dual mode):

1. DAC channel1 with DAC_OUT1 (PA4) as output or connected to on-chip peripherals (ex. OPAMPs, comparators).
2. Whenever present, DAC channel2 with DAC_OUT2 (PA5) as output or connected to on-chip peripherals (ex. OPAMPs, comparators).

### DAC Triggers

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

Digital to Analog conversion can be triggered by:

1. External event: EXTI Line 9 (any GPIOx_PIN_9) using DAC_TRIGGER_EXT_IT9. The used pin (GPIOx_PIN_9) must be configured in input mode.
2. Timers TRGO: TIM2, TIM3, TIM4, TIM5, TIM6 and TIM7 (DAC_TRIGGER_T2_TRGO, DAC_TRIGGER_T3_TRGO...)
3. Software using DAC_TRIGGER_SOFTWARE

### DAC Buffer mode feature

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

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

### DAC connect feature

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

### GPIO configurations guidelines

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

### DAC Sample and Hold feature

For each converter, 2 modes are supported: normal mode and "sample and hold" mode (i.e. low power mode). In the sample and hold mode, the DAC core converts data, then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A new stabilization period is needed before each new conversion. The sample and hold allow setting internal or external voltage @ low power consumption cost (output value can be at any given rate either by CPU or DMA). The Sample and hold block and registers uses either LSI & run in several power modes: run mode, sleep mode, low power run, low power sleep mode & stop1 mode. Low power stop1 mode allows only static conversion. To enable Sample and Hold mode Enable LSI using HAL_RCC_OscConfig with RCC_OSCILLATORTYPE_LSI & RCC_LSI_ON parameters. Use DAC_InitStructure.DAC_SampleAndHold = DAC_SAMPLEANDHOLD_ENABLE; &

DAC_ChannelConfTypeDef.DAC_SampleAndHoldConfig.DAC_SampleTime, DAC_HoldTime & DAC_RefreshTime;

### DAC calibration feature

1. The 2 converters (channel1 & channel2) provide calibration capabilities.
   - Calibration aims at correcting some offset of output buffer.
   - The DAC uses either factory calibration settings OR user defined calibration (trimming) settings (i.e. trimming mode).
   - The user defined settings can be figured out using self calibration handled by HAL_DACEx_SelfCalibrate.
   - HAL_DACEx_SelfCalibrate:
     - Runs automatically the calibration.
     - Enables the user trimming mode
     - Updates a structure with trimming values with fresh calibration results. The user may store the calibration results for larger (ex monitoring the trimming as a function of temperature for instance)

### DAC wave generation feature

Both DAC channels can be used to generate

1. Noise wave
2. Triangle wave

### DAC data format

The DAC data format can be:

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

### DAC data value to voltage correspondence

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

DAC_OUTx = VREF+ * DOR / 4095

- with DOR is the Data Output Register

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

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

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

### DMA requests

A DMA1 request can be generated when an external trigger (but not a software trigger) occurs if DMA1 requests are enabled using HAL_DAC_Start_DMA(). DMA requests are mapped as following:

1. DAC channel1: mapped either on
   - DMA1 request 6 channel3
   - or DMA2 request channel4 which must be already configured
2. DAC channel2 (whenever present): mapped either on
   - DMA1 request 5 channel4
   - or DMA2 request 3 channel5 which must be already configured

For Dual mode and specific signal (Triangle and noise) generation please refer to Extended Features Driver description

### 14.2.2 How to use this driver

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

#### Calibration mode IO operation

- Retrieve the factory trimming (calibration settings) using HAL_DACEx_GetTrimOffset()
- Run the calibration using HAL_DACEx_SelfCalibrate()
- Update the trimming while DAC running using HAL_DACEx_SetUserTrimming()

#### Polling mode IO operation

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

#### DMA mode IO operation

- Start the DAC peripheral using HAL_DAC_Start_DMA(), at this stage the user specify the length of data to be transferred at each end of conversion
- At the middle of data transfer HAL_DAC_ConvHalfCpltCallbackCh1() or HAL_DACEx_ConvHalfCpltCallbackCh2() function is executed and user can add his own code by customization of function pointer HAL_DAC_ConvHalfCpltCallbackCh1() or HAL_DACEx_ConvHalfCpltCallbackCh2()
- At The end of data transfer HAL_DAC_ConvCpltCallbackCh1() or HAL_DACEx_ConvHalfCpltCallbackCh2() function is executed and user can add his own code by customization of function pointer HAL_DAC_ConvCpltCallbackCh1() or HAL_DACEx_ConvHalfCpltCallbackCh2()
- In case of transfer Error, HAL_DAC_ErrorCallbackCh1() function is executed and user can add his own code by customization of function pointer HAL_DAC_ErrorCallbackCh1
- In case of DMA underrun, DAC interruption triggers and execute internal function HAL_DAC_IRQHandler. HAL_DAC_DMAUnderrunCallbackCh1() or HAL_DACEx_DMAUnderrunCallbackCh2() function is executed and user can add his own code by customization of function pointer HAL_DAC_DMAUnderrunCallbackCh1() or HAL_DACEx_DMAUnderrunCallbackCh2() and add his own code by customization of function pointer HAL_DAC_ErrorCallbackCh1()
- Stop the DAC peripheral using HAL_DAC_Stop_DMA()

#### DAC HAL driver macros list

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

- __HAL_DAC_ENABLE: Enable the DAC peripheral
- __HAL_DAC_DISABLE: Disable the DAC peripheral

- __HAL_DAC_CLEAR_FLAG: Clear the DAC's pending flags
- __HAL_DAC_GET_FLAG: Get the selected DAC's flag status

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

### 14.2.3 Initialization and de-initialization functions

This section provides functions allowing to:

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

This section contains the following APIs:

- ***HAL_DAC_Init()***
- ***HAL_DAC_DeInit()***
- ***HAL_DAC_MspInit()***
- ***HAL_DAC_MspDeInit()***

### 14.2.4 IO operation functions

This section provides functions allowing to:

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

This section contains the following APIs:

- ***HAL_DAC_Start()***
- ***HAL_DAC_Stop()***
- ***HAL_DAC_Start_DMA()***
- ***HAL_DAC_Stop_DMA()***
- ***HAL_DAC_IRQHandler()***
- ***HAL_DAC_SetValue()***
- ***HAL_DAC_ConvCpltCallbackCh1()***
- ***HAL_DAC_ConvHalfCpltCallbackCh1()***
- ***HAL_DAC_ErrorCallbackCh1()***
- ***HAL_DAC_DMAUnderrunCallbackCh1()***

### 14.2.5 Peripheral Control functions

This section provides functions allowing to:

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

This section contains the following APIs:

- ***HAL_DAC_GetValue()***
- ***HAL_DAC_ConfigChannel()***

### 14.2.6 Peripheral State and Errors functions

This subsection provides functions allowing to

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

This section contains the following APIs:

- ***HAL_DAC_GetState()***
- ***HAL_DAC_GetError()***

### 14.2.7 Detailed description of functions

#### HAL_DAC_Init

| | |
|---|---|
| Function name | **HAL_StatusTypeDef HAL_DAC_Init (DAC_HandleTypeDef * hdac)** |
| Function description | Initialize the DAC peripheral according to the specified parameters in the DAC_InitStruct and initialize the associated handle. |
| Parameters | • **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. |
| Return values | • **HAL:** status |

#### HAL_DAC_DeInit

| | |
|---|---|
| Function name | **HAL_StatusTypeDef HAL_DAC_DeInit (DAC_HandleTypeDef * hdac)** |
| Function description | Deinitialize the DAC peripheral registers to their default reset values. |
| Parameters | • **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. |
| Return values | • **HAL:** status |

#### HAL_DAC_MspInit

| | |
|---|---|
| Function name | **void HAL_DAC_MspInit (DAC_HandleTypeDef * hdac)** |
| Function description | Initialize the DAC MSP. |
| Parameters | • **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. |
| Return values | • **None:** |

#### HAL_DAC_MspDeInit

| | |
|---|---|
| Function name | **void HAL_DAC_MspDeInit (DAC_HandleTypeDef * hdac)** |
| Function description | DeInitialize the DAC MSP. |
| Parameters | • **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC. |
| Return values | • **None:** |

### HAL_DAC_Start

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

Function description
Enables DAC and starts conversion of channel.

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

Return values
- **HAL:** status

### HAL_DAC_Stop

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

Function description
Disables DAC and stop conversion of channel.

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

Return values
- **HAL:** status

### HAL_DAC_Start_DMA

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

Function description
Enables DAC and starts conversion of channel.

Parameters
- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
- **Channel:** The selected DAC channel. This parameter can be one of the following values:
  - DAC_CHANNEL_1: DAC Channel1 selected
  - DAC_CHANNEL_2: DAC Channel2 selected
- **pData:** The destination peripheral Buffer address.
- **Length:** The length of data to be transferred from memory to DAC peripheral
- **Alignment:** Specifies the data alignment for DAC channel. This parameter can be one of the following values:
  - DAC_ALIGN_8B_R: 8bit right data alignment selected
  - DAC_ALIGN_12B_L: 12bit left data alignment selected
  - DAC_ALIGN_12B_R: 12bit right data alignment selected

Return values
- **HAL:** status

### HAL_DAC_Stop_DMA

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

Function description Disables DAC and stop conversion of channel.

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

Return values
- **HAL:** status

### HAL_DAC_IRQHandler

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

Function description Handles DAC interrupt request This function uses the interruption of DMA underrun.

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

Return values
- **None:**

### HAL_DAC_SetValue

Function name **HAL_StatusTypeDef HAL_DAC_SetValue (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data)**

Function description Set the specified data holding register value for DAC channel.

Parameters
- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
- **Channel:** The selected DAC channel. This parameter can be one of the following values:
  - DAC_CHANNEL_1: DAC Channel1 selected
  - DAC_CHANNEL_2: DAC Channel2 selected
- **Alignment:** Specifies the data alignment. This parameter can be one of the following values:
  - DAC_ALIGN_8B_R: 8bit right data alignment selected
  - DAC_ALIGN_12B_L: 12bit left data alignment selected
  - DAC_ALIGN_12B_R: 12bit right data alignment selected
- **Data:** Data to be loaded in the selected data holding register.

Return values
- **HAL:** status

### HAL_DAC_ConvCpltCallbackCh1

Function name **void HAL_DAC_ConvCpltCallbackCh1 (DAC_HandleTypeDef * hdac)**

Function description Conversion complete callback in non-blocking mode for Channel1.

Parameters
- **hdac:** pointer to a DAC_HandleTypeDef structure that

contains the configuration information for the specified DAC.

Return values
- **None:**

### HAL_DAC_ConvHalfCpltCallbackCh1

Function name **void HAL_DAC_ConvHalfCpltCallbackCh1 (DAC_HandleTypeDef * hdac)**

Function description Conversion half DMA transfer callback in non-blocking mode for Channel1.

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

Return values
- **None:**

### HAL_DAC_ErrorCallbackCh1

Function name **void HAL_DAC_ErrorCallbackCh1 (DAC_HandleTypeDef * hdac)**

Function description Error DAC callback for Channel1.

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

Return values
- **None:**

### HAL_DAC_DMAUnderrunCallbackCh1

Function name **void HAL_DAC_DMAUnderrunCallbackCh1 (DAC_HandleTypeDef * hdac)**

Function description DMA underrun DAC callback for channel1.

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

Return values
- **None:**

### HAL_DAC_GetValue

Function name **uint32_t HAL_DAC_GetValue (DAC_HandleTypeDef * hdac, uint32_t Channel)**

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

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

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

### HAL_DAC_ConfigChannel

Function name **HAL_StatusTypeDef HAL_DAC_ConfigChannel**

**(DAC_HandleTypeDef * hdac, DAC_ChannelConfTypeDef * sConfig, uint32_t Channel)**

Function description: Configures the selected DAC channel.

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

Return values:
- **HAL:** status

#### HAL_DAC_GetState

Function name: **HAL_DAC_StateTypeDef HAL_DAC_GetState (DAC_HandleTypeDef * hdac)**

Function description: return the DAC handle state

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

Return values:
- **HAL:** state

#### HAL_DAC_GetError

Function name: **uint32_t HAL_DAC_GetError (DAC_HandleTypeDef * hdac)**

Function description: Return the DAC error code.

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

Return values:
- **DAC:** Error Code

## 14.3 DAC Firmware driver defines

### 14.3.1 DAC

***DAC Channel selection***

DAC_CHANNEL_1

DAC_CHANNEL_2

***DAC ConnectOnChipPeripheral***

DAC_CHIPCONNECT_DISABLE

DAC_CHIPCONNECT_ENABLE

***DAC data alignment***

DAC_ALIGN_12B_R

DAC_ALIGN_12B_L

DAC_ALIGN_8B_R

***DAC Error Code***

| | |
|---|---|
| HAL_DAC_ERROR_NONE | No error |
| HAL_DAC_ERROR_DMAUNDERRUNCH1 | DAC channel1 DMA underrun error |
| HAL_DAC_ERROR_DMAUNDERRUNCH2 | DAC channel2 DMA underrun error |
| HAL_DAC_ERROR_DMA | DMA error |
| HAL_DAC_ERROR_TIMEOUT | Timeout error |

***DAC Exported Macros***

__HAL_DAC_RESET_HANDLE_STATE

**Description:**

- Reset DAC handle state.

**Parameters:**

- __HANDLE__: specifies the DAC handle.

**Return value:**

- None

__HAL_DAC_ENABLE

**Description:**

- Enable the DAC channel.

**Parameters:**

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

**Return value:**

- None

__HAL_DAC_DISABLE

**Description:**

- Disable the DAC channel.

**Parameters:**

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

**Return value:**

- None

DAC_DHR12R1_ALIGNMENT

**Description:**

- Set DHR12R1 alignment.

**Parameters:**

- __ALIGNMENT__: specifies the DAC alignment

**Return value:**

- None

DAC_DHR12R2_ALIGNMENT

**Description:**

- Set DHR12R2 alignment.

**Parameters:**

- __ALIGNMENT__: specifies the DAC alignment

**Return value:**

- None

DAC_DHR12RD_ALIGNMENT

**Description:**

- Set DHR12RD alignment.

**Parameters:**

- __ALIGNMENT__: specifies the DAC alignment

**Return value:**

- None

__HAL_DAC_ENABLE_IT

**Description:**

- Enable the DAC interrupt.

**Parameters:**

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

**Return value:**

- None

__HAL_DAC_DISABLE_IT

**Description:**

- Disable the DAC interrupt.

**Parameters:**

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

**Return value:**

- None

__HAL_DAC_GET_IT_SOURCE

**Description:**

- Check whether the specified DAC interrupt

source is enabled or not.

**Parameters:**

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

**Return value:**

- State: of interruption (SET or RESET)

__HAL_DAC_GET_FLAG

**Description:**

- Get the selected DAC's flag status.

**Parameters:**

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

**Return value:**

- None

__HAL_DAC_CLEAR_FLAG

**Description:**

- Clear the DAC's flag.

**Parameters:**

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

**Return value:**

- None

***DAC flags definition***

DAC_FLAG_DMAUDR1

DAC_FLAG_DMAUDR2

***DAC high frequency interface mode***

| | |
|---|---|
| DAC_HIGH_FREQUENCY_INTERFACE_MODE_DISABLE | High frequency interface mode |

| | |
|---|---|
| | disabled |
| DAC_HIGH_FREQUENCY_INTERFACE_MODE_ABOVE_80MHZ | High frequency interface mode enabled |
| DAC_HIGH_FREQUENCY_INTERFACE_MODE_AUTOMATIC | High frequency interface mode automatic |

***DAC IT definition***

DAC_IT_DMAUDR1

DAC_IT_DMAUDR2

***DAC output buffer***

DAC_OUTPUTBUFFER_ENABLE

DAC_OUTPUTBUFFER_DISABLE

***DAC power mode***

DAC_SAMPLEANDHOLD_DISABLE

DAC_SAMPLEANDHOLD_ENABLE

***DAC trigger selection***

| | |
|---|---|
| DAC_TRIGGER_NONE | Conversion is automatic once the DAC_DHRxxxx register has been loaded, and not by external trigger |
| DAC_TRIGGER_T1_TRGO | TIM1 TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_T2_TRGO | TIM2 TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_T4_TRGO | TIM1 TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_T5_TRGO | TIM5 TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_T6_TRGO | TIM6 TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_T7_TRGO | TIM7 TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_T8_TRGO | TIM8 TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_T15_TRGO | TIM15 TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_LPTIM1_OUT | LPTIM1 OUT TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_LPTIM2_OUT | LPTIM2 OUT TRGO selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_EXT_IT9 | EXTI Line9 event selected as external conversion trigger for DAC channel |
| DAC_TRIGGER_SOFTWARE | Conversion started by software trigger for DAC channel |

***DAC User Trimming***

| | |
|---|---|
| DAC_TRIMMING_FACTORY | Factory trimming |
| DAC_TRIMMING_USER | User trimming |

# 15 HAL DAC Extension Driver

## 15.1 DACEx Firmware driver API description

### 15.1.1 How to use this driver

- When Dual mode is enabled (i.e. DAC Channel1 and Channel2 are used simultaneously): Use HAL_DACEx_DualGetValue() to get digital data to be converted and use HAL_DACEx_DualSetValue() to set digital value to converted simultaneously in Channel 1 and Channel 2.
- Use HAL_DACEx_TriangleWaveGenerate() to generate Triangle signal.
- Use HAL_DACEx_NoiseWaveGenerate() to generate Noise signal.
- HAL_DACEx_SelfCalibrate to calibrate one DAC channel.
- HAL_DACEx_SetUserTrimming to set user trimming value.
- HAL_DACEx_GetTrimOffset to retrieve trimming value (factory setting after reset, user setting if HAL_DACEx_SetUserTrimming have been used at least one time after reset).

### 15.1.2 Extended features functions

This section provides functions allowing to:

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

This section contains the following APIs:

- ***HAL_DACEx_TriangleWaveGenerate()***
- ***HAL_DACEx_NoiseWaveGenerate()***
- ***HAL_DACEx_DualSetValue()***
- ***HAL_DACEx_ConvCpltCallbackCh2()***
- ***HAL_DACEx_ConvHalfCpltCallbackCh2()***
- ***HAL_DACEx_ErrorCallbackCh2()***
- ***HAL_DACEx_DMAUnderrunCallbackCh2()***
- ***HAL_DACEx_SelfCalibrate()***
- ***HAL_DACEx_SetUserTrimming()***
- ***HAL_DACEx_GetTrimOffset()***

### 15.1.3 Peripheral Control functions

This section provides functions allowing to:

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

This section contains the following APIs:

- ***HAL_DACEx_DualGetValue()***
- ***HAL_DACEx_GetTrimOffset()***

### 15.1.4 Detailed description of functions

#### HAL_DACEx_TriangleWaveGenerate

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

Function description Enable or disable the selected DAC channel wave generation.

Parameters
- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
- **Channel:** The selected DAC channel. This parameter can be one of the following values: DAC_CHANNEL_1 / DAC_CHANNEL_2
- **Amplitude:** Select max triangle amplitude. This parameter can be one of the following values:
  - DAC_TRIANGLEAMPLITUDE_1: Select max triangle amplitude of 1
  - DAC_TRIANGLEAMPLITUDE_3: Select max triangle amplitude of 3
  - DAC_TRIANGLEAMPLITUDE_7: Select max triangle amplitude of 7
  - DAC_TRIANGLEAMPLITUDE_15: Select max triangle amplitude of 15
  - DAC_TRIANGLEAMPLITUDE_31: Select max triangle amplitude of 31
  - DAC_TRIANGLEAMPLITUDE_63: Select max triangle amplitude of 63
  - DAC_TRIANGLEAMPLITUDE_127: Select max triangle amplitude of 127
  - DAC_TRIANGLEAMPLITUDE_255: Select max triangle amplitude of 255
  - DAC_TRIANGLEAMPLITUDE_511: Select max triangle amplitude of 511
  - DAC_TRIANGLEAMPLITUDE_1023: Select max triangle amplitude of 1023
  - DAC_TRIANGLEAMPLITUDE_2047: Select max triangle amplitude of 2047
  - DAC_TRIANGLEAMPLITUDE_4095: Select max triangle amplitude of 4095

Return values
- **HAL:** status

#### HAL_DACEx_NoiseWaveGenerate

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

Function description Enable or disable the selected DAC channel wave generation.

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

- **Amplitude:** Unmask DAC channel LFSR for noise wave generation. This parameter can be one of the following values:
  - DAC_LFSRUNMASK_BIT0: Unmask DAC channel LFSR bit0 for noise wave generation
  - DAC_LFSRUNMASK_BITS1_0: Unmask DAC channel LFSR bit[1:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS2_0: Unmask DAC channel LFSR bit[2:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS3_0: Unmask DAC channel LFSR bit[3:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS4_0: Unmask DAC channel LFSR bit[4:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS5_0: Unmask DAC channel LFSR bit[5:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS6_0: Unmask DAC channel LFSR bit[6:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS7_0: Unmask DAC channel LFSR bit[7:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS8_0: Unmask DAC channel LFSR bit[8:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS9_0: Unmask DAC channel LFSR bit[9:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS10_0: Unmask DAC channel LFSR bit[10:0] for noise wave generation
  - DAC_LFSRUNMASK_BITS11_0: Unmask DAC channel LFSR bit[11:0] for noise wave generation

Return values

- **HAL:** status

### HAL_DACEx_DualSetValue

Function name

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

Function description

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

Parameters

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

Return values

- **HAL:** status

Notes

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

### HAL_DACEx_ConvCpltCallbackCh2

Function name **void HAL_DACEx_ConvCpltCallbackCh2 (DAC_HandleTypeDef * hdac)**

Function description Conversion complete callback in non-blocking mode for Channel2.

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

Return values
- **None:**

### HAL_DACEx_ConvHalfCpltCallbackCh2

Function name **void HAL_DACEx_ConvHalfCpltCallbackCh2 (DAC_HandleTypeDef * hdac)**

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

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

Return values
- **None:**

### HAL_DACEx_ErrorCallbackCh2

Function name **void HAL_DACEx_ErrorCallbackCh2 (DAC_HandleTypeDef * hdac)**

Function description Error DAC callback for Channel2.

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

Return values
- **None:**

### HAL_DACEx_DMAUnderrunCallbackCh2

Function name **void HAL_DACEx_DMAUnderrunCallbackCh2 (DAC_HandleTypeDef * hdac)**

Function description DMA underrun DAC callback for Channel2.

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

Return values
- **None:**

### HAL_DACEx_SelfCalibrate

Function name **HAL_StatusTypeDef HAL_DACEx_SelfCalibrate (DAC_HandleTypeDef * hdac, DAC_ChannelConfTypeDef * sConfig, uint32_t Channel)**

Function description Run the self calibration of one DAC channel.

Parameters
- **hdac:** pointer to a DAC_HandleTypeDef structure that contains the configuration information for the specified DAC.
- **sConfig:** DAC channel configuration structure.
- **Channel:** The selected DAC channel. This parameter can be

one of the following values:
  - DAC_CHANNEL_1: DAC Channel1 selected
  - DAC_CHANNEL_2: DAC Channel2 selected

Return values
- **Updates:** DAC_TrimmingValue. , DAC_UserTrimming set to DAC_UserTrimming
- **HAL:** status

Notes
- Calibration runs about 7 ms.

### HAL_DACEx_SetUserTrimming

Function name **HAL_StatusTypeDef HAL_DACEx_SetUserTrimming (DAC_HandleTypeDef * hdac, DAC_ChannelConfTypeDef * sConfig, uint32_t Channel, uint32_t NewTrimmingValue)**

Function description Set the trimming mode and trimming value (user trimming mode applied).

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

Return values
- **HAL:** status

### HAL_DACEx_DualGetValue

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

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

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

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

### HAL_DACEx_GetTrimOffset

Function name **uint32_t HAL_DACEx_GetTrimOffset (DAC_HandleTypeDef * hdac, uint32_t Channel)**

Function description Return the DAC trimming value.

Parameters
- **hdac::** DAC handle
- **Channel:** The selected DAC channel. This parameter can be one of the following values:
  - DAC_CHANNEL_1: DAC Channel1 selected
  - DAC_CHANNEL_2: DAC Channel2 selected

Return values
- **Trimming:** value: range: 0->31

#### DAC_DMAConvCpltCh2

| | |
|---|---|
| Function name | **void DAC_DMAConvCpltCh2 (DMA_HandleTypeDef * hdma)** |
| Function description | DMA conversion complete callback. |
| Parameters | • **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module. |
| Return values | • **None:** |

#### DAC_DMAErrorCh2

| | |
|---|---|
| Function name | **void DAC_DMAErrorCh2 (DMA_HandleTypeDef * hdma)** |
| Function description | DMA error callback. |
| Parameters | • **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module. |
| Return values | • **None:** |

#### DAC_DMAHalfConvCpltCh2

| | |
|---|---|
| Function name | **void DAC_DMAHalfConvCpltCh2 (DMA_HandleTypeDef * hdma)** |
| Function description | DMA half transfer complete callback. |
| Parameters | • **hdma:** pointer to a DMA_HandleTypeDef structure that contains the configuration information for the specified DMA module. |
| Return values | • **None:** |

## 15.2 DACEx Firmware driver defines

### 15.2.1 DACEx

***DACEx lfsrunmask triangle amplitude***

| | |
|---|---|
| DAC_LFSRUNMASK_BIT0 | Unmask DAC channel LFSR bit0 for noise wave generation |
| DAC_LFSRUNMASK_BITS1_0 | Unmask DAC channel LFSR bit[1:0] for noise wave generation |
| DAC_LFSRUNMASK_BITS2_0 | Unmask DAC channel LFSR bit[2:0] for noise wave generation |
| DAC_LFSRUNMASK_BITS3_0 | Unmask DAC channel LFSR bit[3:0] for noise wave generation |
| DAC_LFSRUNMASK_BITS4_0 | Unmask DAC channel LFSR bit[4:0] for noise wave generation |
| DAC_LFSRUNMASK_BITS5_0 | Unmask DAC channel LFSR bit[5:0] for noise wave generation |
| DAC_LFSRUNMASK_BITS6_0 | Unmask DAC channel LFSR bit[6:0] for noise wave |