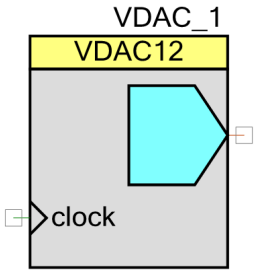
**Please note that Cypress is an Infineon Technologies Company.**
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

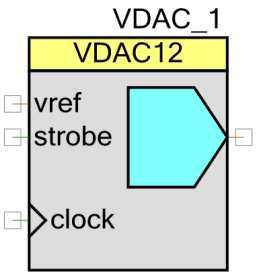
**Continuity of document content**
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

**Continuity of ordering part numbers**
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

# Voltage DAC (12-bit) (VDAC12_PDL)
1.0

## Features




- 12-bit continuous-time DAC (CTDAC)
- Selectable voltage references:
  - Analog supply voltage (VDDA)
  - Buffered internal analog reference
  - Buffered external reference
- Selectable output paths:
  - Direct DAC output to a pin or to an internal Component
  - Buffered DAC output to a pin or to an internal Component
- Selectable input modes:
  - 12-bit unsigned mode
  - 12-bit two's-complement mode
- Optional sample and hold circuit connection for low power Deep Sleep operation
- 500 ksps maximum programmable update rate
- Interrupt and DMA trigger on DAC buffer empty
- Can be enabled (but not updated) in Deep Sleep power mode

## General Description

The VDAC12_PDL is a 12-bit DAC based on a resistor ladder DAC. The core is closely integrated with the Continuous Time Block (CTB), which provides easy buffering of the DAC output voltage through one of its opamps. The CTB can additionally provide a buffered input reference voltage and a sample/hold feature for reduced power while maintaining the DAC output. See the Technical Reference Manual (TRM) for details on the CTB. The DAC reference can come from VDDA or from any buffered signal though the CTB opamp. This can be an external signal through a GPIO or from the analog reference (AREF). The control interface provides an option to control the DAC output through CPU and DMA. This includes a double buffered DAC voltage control register, clock input for programmable update rate, interrupt on DAC buffer empty to CPU, and trigger to DMA.

## Quick Start

1. Drag an VDAC12_PDL Component from the Component Catalog Cypress/Analog/DAC folder onto your schematic (placed instance takes the name **VDAC_1**).
2. Double-click to open the Configure dialog.
3. Set up the desired settings (Vref Source, Output Buffer, DAC Mode, etc.). Rename the instance name to **VDAC** for readability.
4. Connect the input/output terminals (see Input/Output Connections).
5. Open the Design-Wide Resources Pin Editor and assign the input and output pins for your design.
6. Build the project in order to verify the correctness of your design. This will add the required PDL modules to the Workspace Explorer, and generate configuration data for the **VDAC** instance.
7. In the *main.c* file, initialize the peripheral and start the application:

```
/* Initialize the CTDAC peripheral. */
(void)Cy_CTDAC_Init(VDAC_CTDAC_HW, &VDAC_ctdacConfig);
#if VDAC_PARAM_ReferenceBufferUsed
    /* Initialize the opamp used by the reference buffer, if a reference buffer is selected in the
   customizer GUI. */
    (void)Cy_CTB_OpampInit(VDAC_CTB_HW, VDAC_CTB_REFBUF_OPAMP_NUM,
  &VDAC_refBufferConfig);
#endif
#if VDAC_PARAM_OutputBufferUsed
    /* Initialize the opamp used by the output buffer, if an output buffer is selected in the
   customizer GUI. */
    (void)Cy_CTB_OpampInit(VDAC_CTB_HW,
  VDAC_CTB_VOUTBUF_OPAMP_NUM, &VDAC_outputBufferConfig);
#endif

/* Enable the CTDAC peripheral. */
Cy_CTDAC_Enable(VDAC_CTDAC_HW);
#if (VDAC_PARAM_ReferenceBufferUsed || VDAC_PARAM_OutputBufferUsed)
    /* Enable the CTB peripheral if at least one buffer is used. */
    Cy_CTB_Enable(VDAC_CTB_HW);
#endif
```

8. Build the project and program the device.

# Input/Output Connections

This section describes the various input and output connections for the VDAC12_PDL Component. An asterisk (∗) after the terminal name indicates that the terminal may not be shown on the Component symbol for the conditions listed in the description of that I/O terminal.

| Terminal | I/O Type | Description |
|---|---|---|
| DAC output | Analog Output | This is the DAC output terminal. This terminal is always present. |
| trigger∗ | Digital Output | This output is a trigger signal when a VDAC value update is completed. The trigger signal is high for two PeriClk cycles. The terminal appears when the Show Trigger Output Terminal ∗ check box is checked and the Update Mode is not Direct write. |

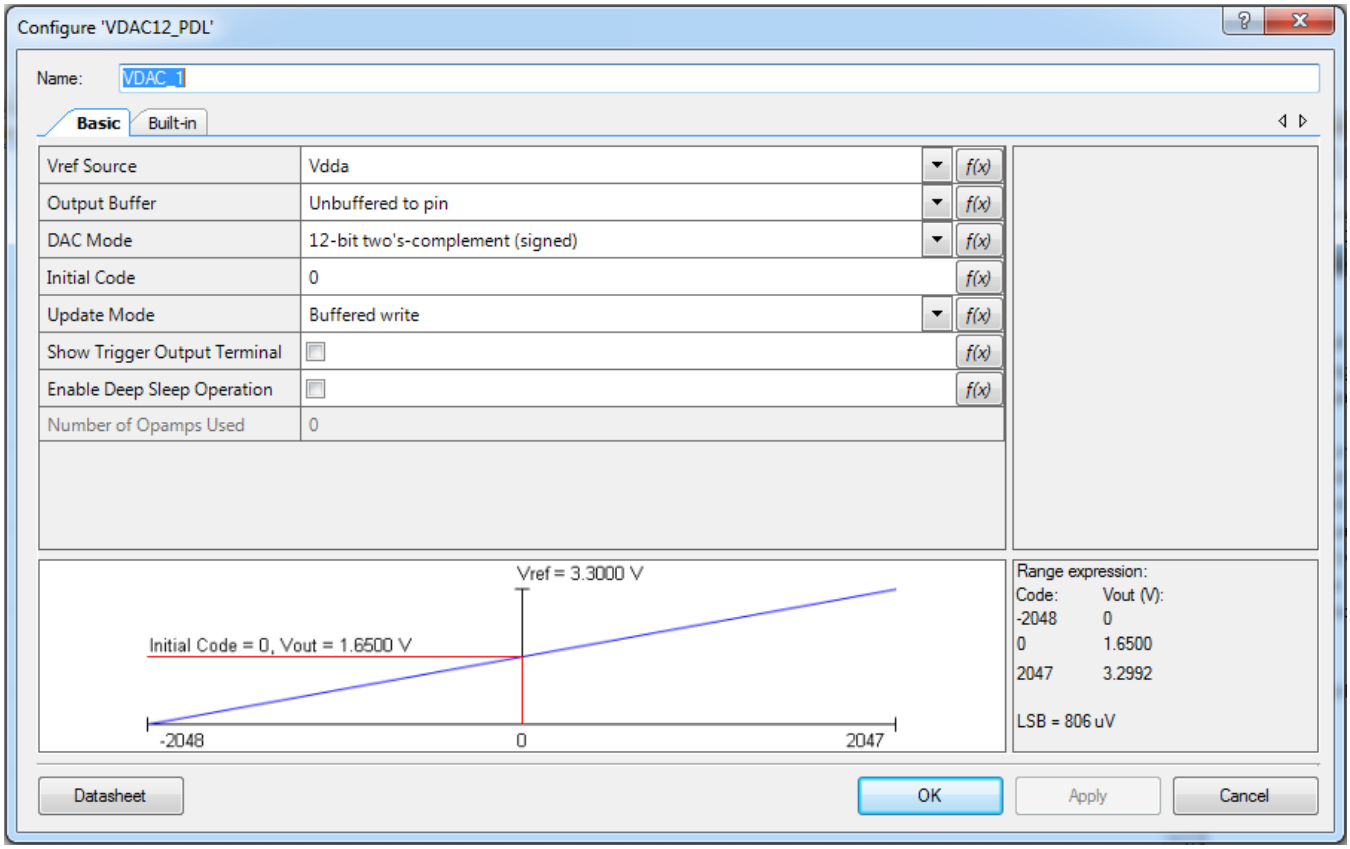
| Terminal | I/O Type | Description |
|---|---|---|
| vref∗ | Analog Input | When the reference source is external, this terminal will appear. This reference will be buffered using Opamp1 of the CTB. |
| clock∗ | Digital Input | This terminal is for a user defined clock input. It appears when the Update Mode is Buffered write, Strobe edge sync, or Strobe level. The maximum clock frequency is 500 kHz. |
| strobe∗ | Digital Input | This terminal is for a user defined strobe input. It appears when the Update Mode is Strobe edge sync, Strobe edge immediate, or Strobe level. |

# Component Parameters

This section covers the various parameters available from the Component's Configure dialog. Drag a VDAC12_PDL Component from the Component Catalog onto your schematic and double-click it to open the dialog.

For any selectable parameter, the option shown here in **bold** is the default. An asterisk (∗) near the parameter name indicates that the parameter will appear only under certain conditions.

## Basic Tab



### Vref Source

The voltage reference source selection.

- External
- **Vdda**
- Analog reference

When External or Analog reference is selected, an opamp will be used to buffer the reference. When a buffer is used, the voltage range will be reduced (see Opamp Input and Output Ranges).

The analog reference can be found in the Design-Wide Resources System Editor. For this Component, use either:

- Local (1.2 V): when using the on-chip generated reference.
- External Pin: when using an off-chip reference.

### Vref Voltage (V) *

This parameter is used to specify the Vref voltage. It determines the range expression and transfer graph displayed in the GUI. It is also provided as a constant in millivolts for run-time conversion of counts to volts. It is only visible when Vref Source is set to External or Analog reference with the Bandgap Value of the Analog reference set to External Pin.

The default value is **1.200** V. The VDAC reference voltage should be at least 1.200 V and no greater than VDDA. VDAC performance is not guaranteed with references less than 1.200 V.

### Output Buffer

The output buffer selection.

- **Unbuffered to pin**
- Unbuffered internal
- Buffered to pin
- Buffered internal

In Unbuffered to pin mode, the output is routed directly to Pin 6 of the VDAC dedicated port. See the device datasheet for this port. See the I/O System section of the device TRM for a description of ports.

In Unbuffered internal mode, the output is routed through the internal switches of the CTB to OA0. In this mode, OA0 is unused (not configured as a buffer) so that it can be used as desired in the design (for example, as a comparator).

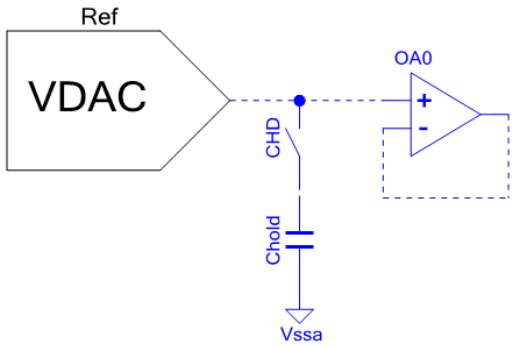
In Buffered to pin mode, the output is buffered and intended to be sent to an external pin.

In Buffered internal mode, the output is buffered and intended to be sent to an internal component. A build error will occur if the VDAC output is routed to an external pin in this mode. The differentiation between internal and external is needed in order to configure the opamp for sufficient drive strength.

When a buffer is used, the output voltage range will be reduced (see Opamp Input and Output Ranges).

### Sample and Hold *

This check box is visible when the output is buffered. When checked, the sample and hold capacitor in the CTB hardware block will be connected to the CTDAC output by closing the CHD switch. Use the VDAC_EasySampleAndHold() function to perform the sample and hold routines.




### DAC Mode

Format selection for the relationship between the code and output voltage. This can be changed during runtime with the Cy_CTDAC_SetSignMode() function.

- 12-bit unsigned
- **12-bit two's complement (signed)**

The following range expressions are based on Unbuffered modes. Recall that when a buffer is used, the voltage range will be reduced (see Opamp Input and Output Ranges).

| DAC mode | LSB | Range |
|---|---|---|
| 12-bit unsigned | Vref/4096 | Code 0: Vout = 0<br>Code 2048: Vout = 0.5 * Vref<br>Code 4095: Vout = Vref * 4095 / 4096 |
| 12-bit two's-complement (signed) | Vref/4096 | Code -2048: Vout = 0<br>Code 0: Vout = 0.5 * Vref<br>Code 2047: Vout = Vref * 4095 / 4096 |

### Initial Code

User entry for setting the initial code after the Component is initialized and enabled. The supported range is -2048 to 2047 for signed and 0 to 4095 for unsigned. Values outside the supported range are not permitted.

The default value is **0**.

### Update Mode

VDAC update mode selection. Sets when and how the VDAC output is updated.

| Update mode | strobe? | clock? | API | Description |
|---|---|---|---|---|
| Direct write | No | No | Cy_CTDAC_SetValue() | VDAC value is updated when data is written into the CTDAC_VAL register |
| **Buffered write** | No | Yes | Cy_CTDAC_SetValueBuffered() | VDAC value is updated on every VDAC clock rising edge |
| Strobe edge sync | Yes | Yes | Cy_CTDAC_SetValueBuffered() | VDAC value is updated on the first VDAC clock rising edge after a strobe rising edge |
| Strobe edge immediate | Yes | No | Cy_CTDAC_SetValueBuffered() | VDAC value is updated on every strobe rising edge |
| Strobe level | Yes | Yes | Cy_CTDAC_SetValueBuffered() | VDAC value is updated on every VDAC clock rising edge while strobe is high |

The high time of the strobe signal must be at least two PeriClk cycles. The low time of the strobe signal must also be at least two PeriClk cycles. This limits the maximum strobe frequency to $\frac{PeriClkFrequency}{4}$. See Update Mode Timing Diagrams for more information.

Calling the wrong API function in the selected update mode would result in unexpected behavior. In Direct write, update the VDAC output by calling Cy_CTDAC_SetValue(). Calling Cy_CTDAC_SetValueBuffered() in this mode would never update the VDAC output. In all other update modes, update the VDAC output by calling Cy_CTDAC_SetValueBuffered(). Calling Cy_CTDAC_SetValue() in these modes could update the VDAC output momentarily but would be overwritten on the next VDAC clock or strobe edge with the value in the buffered register.

### Show Trigger Output Terminal *

Check box to show the trigger output terminal on the symbol. This check box is not visible when the Update Mode is Direct write.

The trigger signal is high for two PeriClk cycles and it can be used with a DMA Component to initiate code updates.

### Enable Deep Sleep Operation

Check box to enable the CTDAC and Opamp hardware blocks in Deep Sleep mode. It also ensures that only Deep Sleep capable switches are used for analog routing during build time and that the Analog Reference (AREF) block that supplies the reference currents for the opamp is also enabled in Deep Sleep mode.

If checked, the input range for all used opamps is 0 V to VDDA - 1.5 V in all power modes (Active, Low Power, and Deep Sleep). See Opamp Input and Output Ranges for more info.

The Cy_CTDAC_SetDeepSleepMode() function can be used during runtime to disable operation in Deep Sleep mode. However, this function should only be used if this check box is checked.

In Deep Sleep mode, all clocks on the device are disabled and the VDAC output cannot be updated. The VDAC maintains the last code value before entering Deep Sleep.

If the Vref Source is set to Internal analog reference, the check box in the Design-Wide Resources System Editor to enable the reference during Deep Sleep must also be selected.

The Component has a Deep Sleep callback function that must be registered with the Cy_SysPm_-Registercallback() function. This callback guarantees that the Component is prepared for Deep Sleep entry and ready after wakeup. (see VDAC_DeepSleepCallbackStruct).

### Buffer Input Range (display only) *

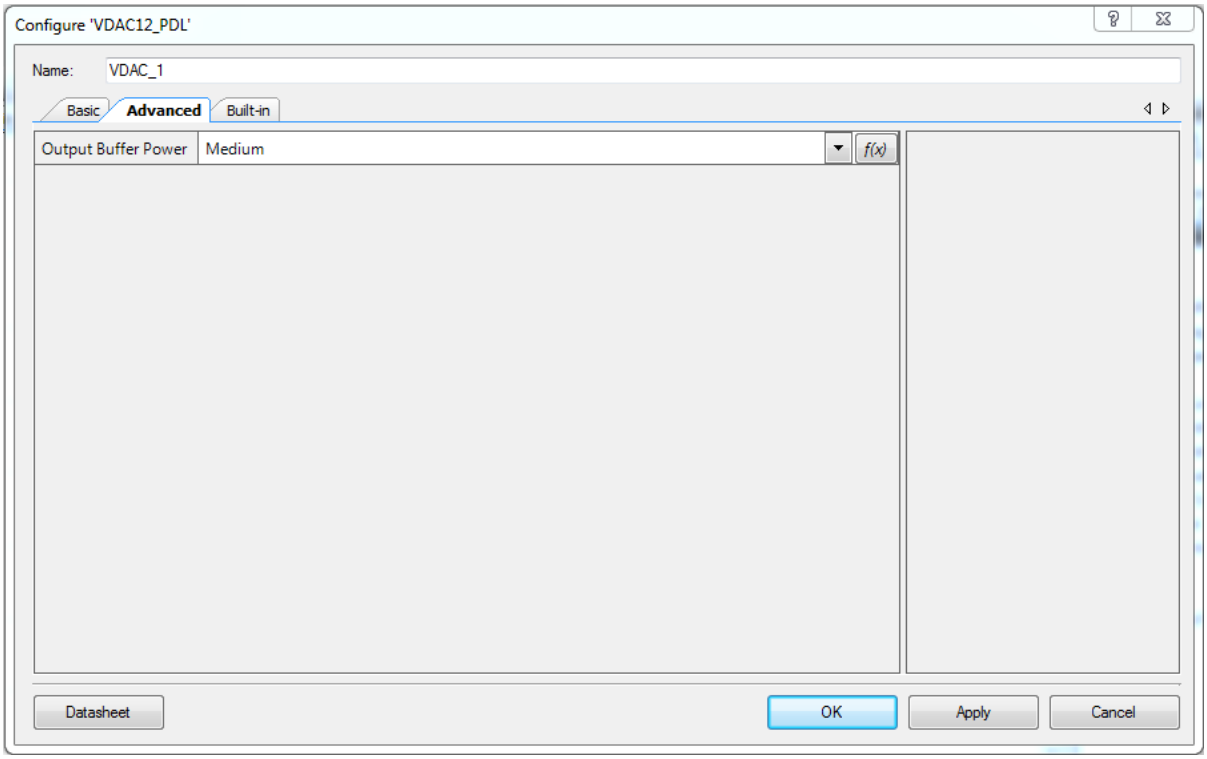
If any opamps are used, this displays the opamp input range. The input range depends on the Enable Deep Sleep Operation selection and the opamp reference current level selection. If Deep Sleep is enabled or the opamp reference current level is set to 100 nA, the opamp charge pumps are disabled and the input range is limited to 0 V to VDDA - 1.5 V. When the charge pump is enabled, the input range is rail-to-rail.

### Number of Opamps Used (display only)

Displays the number of opamps that are used based on the reference source and output buffer selections. The minimum value is zero and the maximum value is two.

## Advanced Tab *

This tab is visible only when the Output Buffer is Buffered to Pin or Buffered internal. There is only one user selection in this tab.




### Output Buffer Power

Power selection of the output buffer.

- Low
- **Medium**
- High

Medium is the minimum power level to achieve the VDAC performance specifications. The low power level can be used to save power but with reduced VDAC performance. The high power level can be used for higher current drive.

The power setting will change the power consumption and gain bandwidth of the output driving buffer. The power consumption and gain bandwidth are also affected by the Opamp Reference Current (1 uA or 100 nA) as set by the user in the Design-Wide Resources System Editor and the device power state. Refer to the device datasheet for the opamp specifications.

# Application Programming Interface

The Application Programming Interface (API) is provided by the cy_ctdac driver module from the PDL. The driver is copied into the “pdl\” directory of the application project after a successful build.

Refer to the PDL documentation for a detailed description of the complete API. To access this document, right-click on the Component symbol on the schematic and choose the **“Open PDL Documentation. . . ”** option in the drop-down menu.

The Component generates the configuration structures and base address described in the Global Variables section. Pass the generated data structure and the base address to the associated cy_ctdac driver function in the application initialization code to configure the peripheral. Once the peripheral is initialized, the application code can perform run-time changes by referencing the provided base address in the driver API functions.

By default, PSoC Creator assigns the instance name **VDAC_1** to the first instance of the VDAC12_PDL in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following section is **VDAC**.

## Global Variables

The VDAC12_PDL Component populates the following Deep Sleep callback and peripheral initialization data structures.

### Variables

- cy_stc_syspm_callback_t VDAC_DeepSleepCallbackStruct
- const cy_stc_ctdac_config_t VDAC_ctdacConfig
- const cy_stc_ctb_opamp_config_t VDAC_outputBufferConfig
- const cy_stc_ctb_opamp_config_t VDAC_refBufferConfig

### Variable Documentation

*cy_stc_syspm_callback_t VDAC_DeepSleepCallbackStruct*

System power management callback structure for Deep Sleep entry and exit. This callback is needed in order to disable deglitching before the device enters Deep Sleep. This ensures that the deglitch switches are closed and that DAC operation in Deep Sleep, if enabled, is correct. The callback also ensures that deglitching is re-enabled when the device wakes up. Registering this Deep Sleep callback is needed only if Deep Sleep operation is enabled in the Customizer Configure dialog (see Enable Deep Sleep Operation).

```
/* Register Component callback. */
Cy_SysPm_RegisterCallback(&VDAC_DeepSleepCallbackStruct);

/* Put device into Deep Sleep. */
Cy_SysPm_DeepSleep(CY_SYSPM_WAIT_FOR_INTERRUPT);
```

*const cy_stc_ctdac_config_t VDAC_ctdacConfig*

Configuration structure for initializing the CTDAC PDL

*const cy_stc_ctb_opamp_config_t VDAC_outputBufferConfig*

Configuration structure for initializing Opamp0 of the CTB as the CTDAC output buffer. Defined only if the output is buffered.

*const cy_stc_ctb_opamp_config_t VDAC_refBufferConfig*

Configuration structure for initializing Opamp1 of the CTB as the CTDAC reference buffer. Defined only if the references is not Vdda.

## Preprocessor Macros

The VDAC12_PDL Component generates the following preprocessor macros.

### Macros

- #define VDAC_CTDAC_HW CTDAC0
- #define VDAC_CTB_HW CTB0
- #define VDAC_CTB_VOUTBUF_OPAMP_NUM CY_CTB_OPAMP_0
- #define VDAC_CTB_REFBUF_OPAMP_NUM CY_CTB_OPAMP_1

### Macro Definition Documentation

*#define VDAC_CTDAC_HW CTDAC0*

The pointer to the base address of the CTDAC instance

*#define VDAC_CTB_HW CTB0*

The pointer to the base address of the CTB instance. Defined only if buffers are used.

*#define VDAC_CTB_VOUTBUF_OPAMP_NUM CY_CTB_OPAMP_0*

The CTB Opamp number associated to the output buffer. Defined only if the output is buffered.

*#define VDAC_CTB_REFBUF_OPAMP_NUM CY_CTB_OPAMP_1*

The CTB Opamp number associated to the reference buffer. Defined only if the reference is not Vdda.

## Component Functions

The VDAC12_PDL Component provides the following function for performing Sample and Hold. The Component also includes a set of Component-specifc wrapper functions that provide simplified access to the low-level functions in the CTDAC PDL module.

### Functions

- void VDAC_EasySampleAndHold (void)

### Function Documentation

*void VDAC_EasySampleAndHold ( void )*

Performs entire sequence of API functions to sample and hold the DAC output.

1. If DAC hardware is not enabled,

(a) Enable DAC hardware
(b) Wait 2 us for DAC hardware to start up

2. Perform sample by calling VDAC_SetSampleAndHold() with VDAC_SH_SAMPLE
3. Wait 10 us to allow for hold cap to charge
4. Perform hold by calling VDAC_SetSampleAndHold() with VDAC_SH_HOLD

This function is useful only when the output is buffered and the sample and hold feature is enabled in the customizer.

This function should be called periodically to refresh the voltage on the hold capacitor. The hold time depends on the supply and reference voltages. The following hold times are based on the time it takes for the opamp output to change by 1 LSB.

- Hold time = 750 us @ Vref = VDDA , VDDA = 1.7 V
- Hold time = 525 us @ Vref = VDDA , VDDA = 3.6 V
- Hold time = 200 us @ Vref = 1.2 V, VDDA = 3.6 V

To hold the DAC output in Deep Sleep mode, the DAC and the output buffer opamp must be enabled for Deep Sleep operation. The Enable Deep Sleep Operation checkbox in the customizer should be enabled during build time to ensure the opamp and the routing will be available in Deep Sleep mode.

This function does not manage the VDAC reference buffer opamp, if used. This is left to the user. The reference opamp can be powered down in Deep Sleep for additional power savings. If powered down, the wake up time of the opamp will need to be considered. The opamp wakeup time can range from 1 us (without charge pump) to 25 us (with charge pump). By default, the reference buffer opamp is configured for Medium power.

**Note** the opamps use reference currents from the Analog Reference (AREF) block. Refer to the TRM for the AREF startup times.

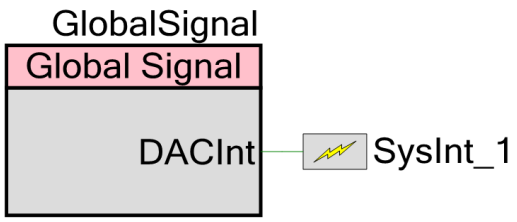
**Note** the sample and hold switches are enabled by the DAC hardware. The DAC hardware should remain enabled during any sample and hold modes.

## Data in Ram

The generated data may be placed in flash memory (const) or RAM. The former is the more memory-efficient choice if you do not wish to modify the configuration data at run-time. Under the Built-In tab of the Configure dialog, enable or disable the "Config Data in Flash" checkbox to make your selection. The default option is to place the data in flash.

## Interrupt Service Routine

When the value is transferred from the CTDAC_VAL_NXT register to the CTDAC_VAL register, an interrupt is generated. This interrupt is global for all CTDACs on the device. Therefore to access the VDAC interrupt, place the Global Signal Reference Component onto the schematic and configure it for the "Combined CT-DAC interrupt (DACInt)" interrupt. The interrupt can be used by the CPU to transfer the next value to the CTDAC_VAL_NXT register.




As with any interrupt, initialize and enable the interrupt and assign the user routine to be called.

```
    (void)Cy_SysInt_Init(&SysInt_1_cfg, CTDAC_ISR);
    NVIC_EnableIRQ(SysInt_1_cfg.intrSrc);
```

The user routine should check for the interrupt status so that if multiple CTDACs are available and enabled on the device, the expected interrupt is handled. For the hardware to generate subsequent interrupts, the user routine must clear the interrupt. The ARM CM0+ and CM4 CPUs use bufferable write transfers to the peripherals by default. See the ARM website for more information.

```
int32_t nextValue;

void CTDAC_ISR(void)
{
    uint8_t status;

    /* Check for interrupt status. */
    status = Cy_CTDAC_GetInterruptStatus(VDAC_CTDAC_HW);

    if (status)
    {
        /* Clear the interrupt. */
        Cy_CTDAC_ClearInterrupt(VDAC_CTDAC_HW);

        /* Add code to execute when an interrupt occurs such as
         * setting the next DAC code value. */
        Cy_CTDAC_SetValueBuffered(VDAC_CTDAC_HW, nextValue);
    }
}
```

## Code Examples and Applications

### Code Examples

PSoC Creator provides access to code examples in the Code Example dialog. For Component-specific examples, open the dialog from the Component Catalog or an instance of the Component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Code Example" topic in the PSoC Creator Help for more information.

There are also numerous code examples that include schematics and example code available online at the Cypress Code Examples web page.

### Application Notes

Cypress provides a number of application notes describing how PSoC can be integrated into your design. You can access the Cypress Application Notes search web page.

# Functional Description

The function of the VDAC12_PDL Component is to produce a steady or time-varying voltage that can be set with 12-bit resolution using a digital code that arrives in an input register by CPU or DMA transfer.

The VDAC12_PDL has buffered and unbuffered output voltage modes. In unbuffered mode the output has a code-independent output resistance of ~15 kohm and can produce output voltages that go to Vssa and (if Vdda is chosen as the reference) to Vdda. In buffered mode, a CTB amplifier implements a unity-gain follower to provide a low output impedance. Closest approach to the supply rails is then determined by the loading on the CTB output.

The VDAC reference voltage can come directly from VDDA, from the internal analog reference, or from another analog signal. When the reference is not VDDA, the source is buffered through an additional CTB amplifier.

## Deglitch

The hardware supports a deglitch mode that reduces the glitch experienced at major code transitions. Deglitching is implemented by controlling a switch on the output path. The switch is opened before the VDAC value is updated. The switch remains open for a configurable number of PeriClk cycles, DEGLITCH_CNT. The deglitch time can be calculated as $\frac{DEGLITCH_CNT+1}{PeriClkFrequency}$.

The VDAC12_PDL Component enables deglitch by default and sets the deglitch time to 700 ns.

## Sample and Hold

The buffered version supports a sample and hold feature that can be used for saving power. The CTDAC output voltage is retained on an internal capacitor for a duration of time while the CTDAC output can be turned off. The CTDAC output needs to be turned on periodically to recharge the hold capacitor. This feature is firmware controlled with the help of the VDAC_EasySampleAndHold() function.

The hold time depends on the supply and reference voltages. The following hold times are based on the time it takes for the buffered output to change by 1 LSB.

- Hold time = 750 us @ Vref = VDDA , VDDA = 1.7 V
- Hold time = 525 us @ Vref = VDDA , VDDA = 3.6 V

- Hold time = 200 us @ Vref = 1.2 V, VDDA = 3.6 V

The sample and hold circuit is contained in the CTB; therefore, it is only available when the output is buffered.

## Opamp Input and Output Ranges

When a buffer is used for the VDAC reference or output, the following opamp input and output ranges should be considered.

The output range of the buffer is 0.2 V to VDDA - 0.2 V (see device datasheet for the effect of output load). The input range of the buffer is 0 V to VDDA, with charge pump enabled, and 0 V to VDDA - 1.5 V, with charge pump disabled. The charge pump is disabled for all power modes (Active, Low Power, and Deep Sleep) if the Enable Deep Sleep Operation check box is checked.

| Charge Pump | Input Range | Output Range | Overall Range |
|---|---|---|---|
| Enabled | 0 V to VDDA | 0.2 V to VDDA - 0.2 V | 0.2 V to VDDA - 0.2 V |
| Disabled | 0 V to VDDA - 1.5 V | 0.2 V to VDDA - 0.2 V | 0.2 V to VDDA - 1.5 V |

## Opamp Ultra Low Power

The buffers can operate in an ultra low power mode by setting the Opamp Reference Current in the Design-Wide Resources System Editor to 100 nA with a gain bandwidth trade-off. This selection in the Design-Wide Resources will affect all opamps on the device. Refer to the device datasheet for the opamp specifications.

## Clocking System

The VDAC is clocked by the PeriClk, which is the source for all programmable peripheral blocks. The PeriClk is configured in Clocks tab of the Design-Wide Resources Clock Editor. See the device TRM for more detail of the device clocking system.

## Update Mode Timing Diagrams

The CTDAC contains two registers:

1. CTDAC_VAL

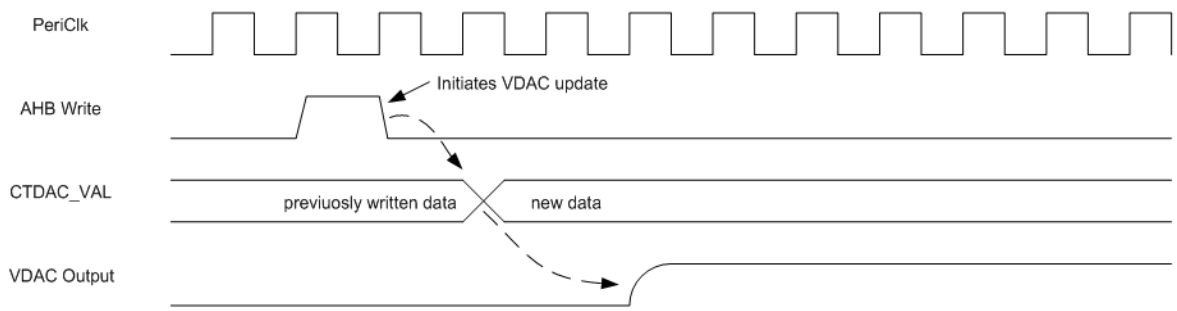
   For direct firmware writes to update the current VDAC value. This register is written with Cy_CTDAC-_SetValue().

2. CTDAC_VAL_NXT For buffered writes to update the VDAC at a periodic rate or on a strobe trigger input. This register is written with Cy_CTDAC_SetValueBuffered().

### Direct Write

In this mode, the user writes directly into the CTDAC_VAL register using Cy_CTDAC_SetValue(). The action of writing to this register will update the VDAC output. This mode does not generate an interrupt or trigger signal and does not require a VDAC clock input.

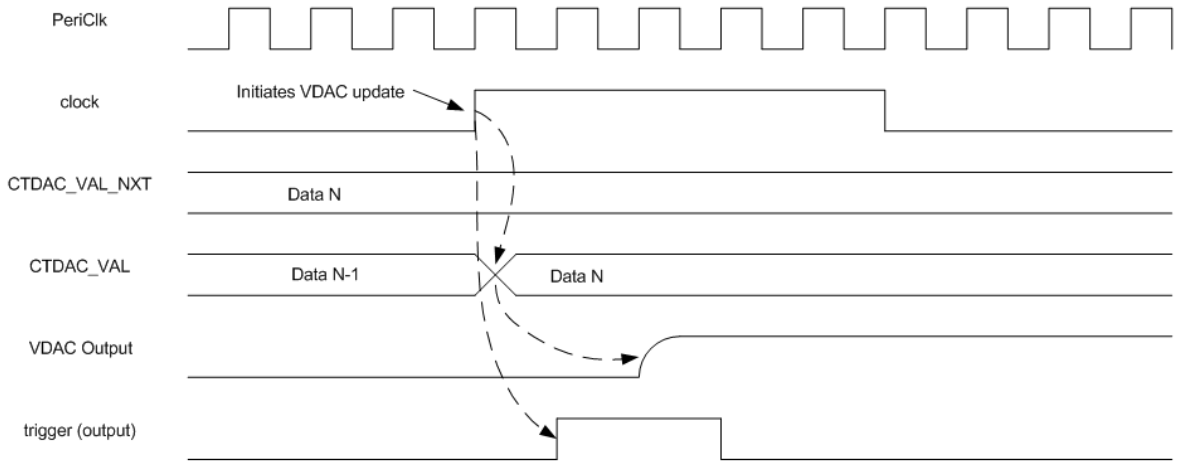
Additionally, calling Cy_CTDAC_SetValueBuffered() does not update the VDAC output in this mode.




## Buffered Write

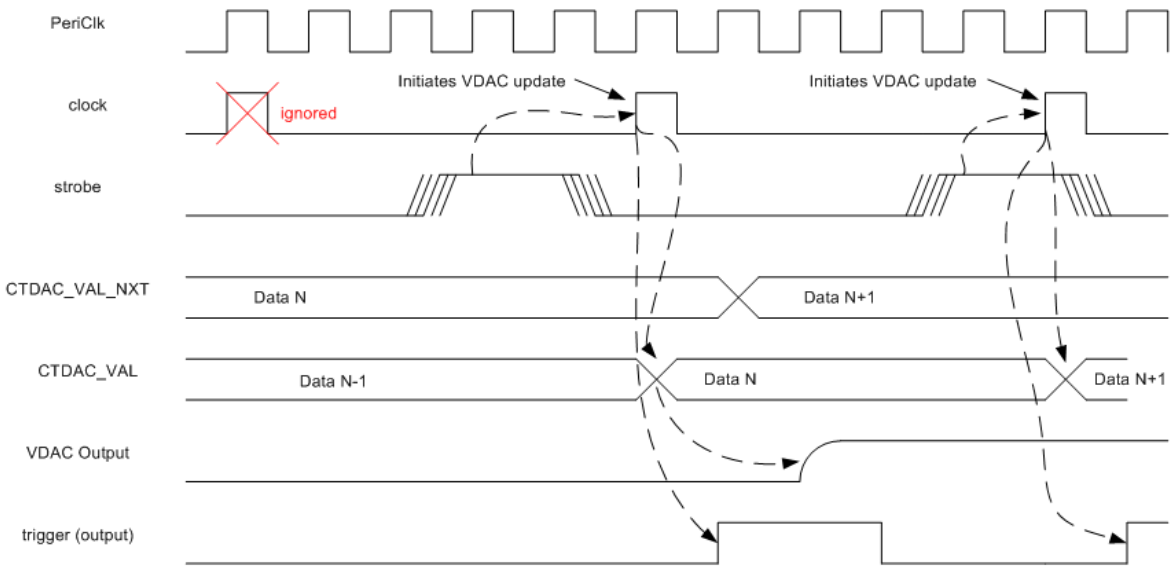
In this mode, the user writes to the CTDAC_VAL_NXT register using Cy_CTDAC_SetValueBuffered(). The rising edge of the VDAC clock will update the VDAC output and generate the interrupt and trigger signal.

Whenever data is transferred from the CTDAC_VAL_NXT register, an interrupt is asserted the same time as the trigger. But while the trigger is automatically cleared after two PeriClk cycles, the user must clear the interrupt with Cy_CTDAC_ClearInterrupt().



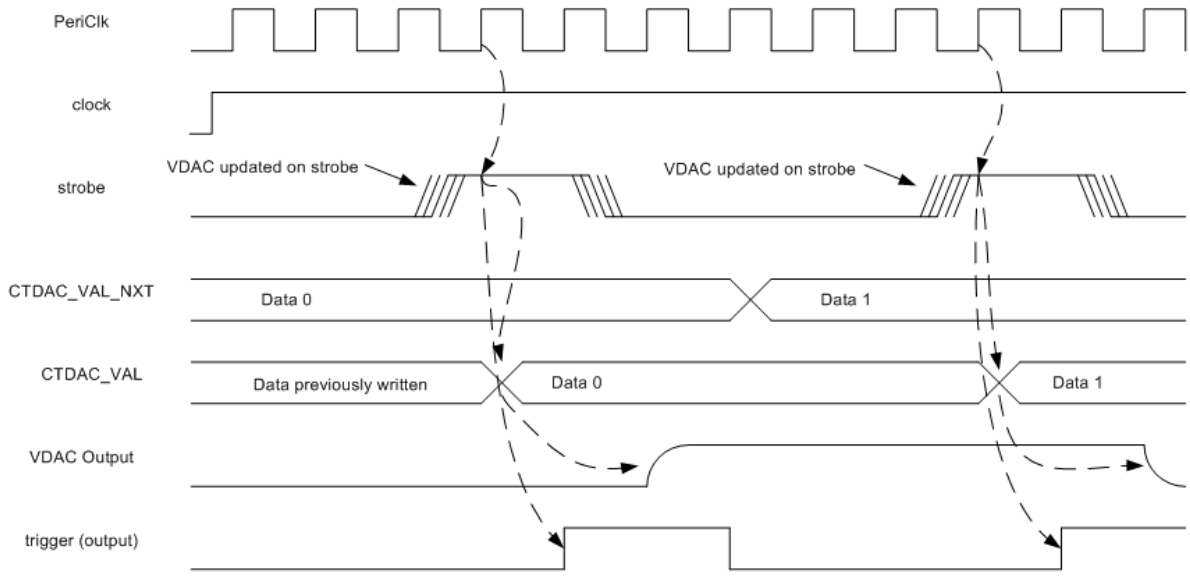
## Strobe Edge Sync

In this mode, the user writes to the CTDAC_VAL_NXT register using Cy_CTDAC_SetValueBuffered(). Each rising edge of the strobe input enables one subsequent update from the rising edge of the clock. The strobe input must remain high for two PeriClk cycles and go low for another two PeriClk cycles to allow for the next update.



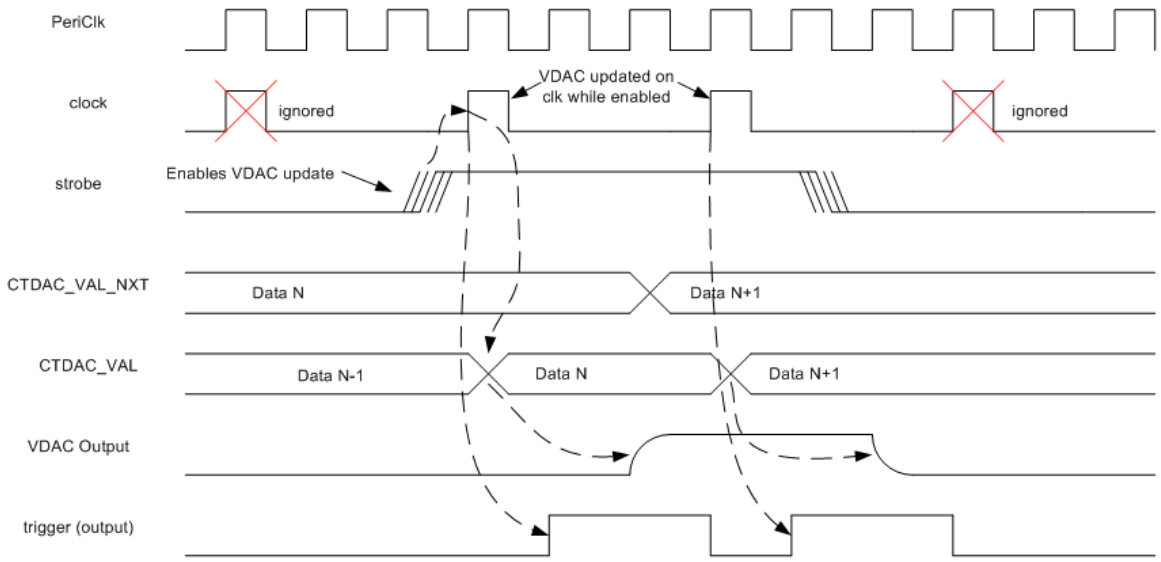

## Strobe Edge Immediate

In this mode, the user writes to the CTDAC_VAL_NXT register using Cy_CTDAC_SetValueBuffered(). The clock resource is used but set to a logic high. Therefore each rising edge of the DSI strobe input immediately updates the VDAC output.



### Strobe Level

In this mode, the user writes to the CTDAC_VAL_NXT register using Cy_CTDAC_SetValueBuffered(). The DSI strobe input acts as a hardware enable signal. While the DSI strobe input is high, the mode behaves like the Buffered write mode. When the DSI strobe input is low, updates are disabled.




# MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations defined:

| Deviation Type | Description |
|---|---|
| Project deviations | Deviations that are applicable for all PSoC Creator Components |
| Specific deviations | Deviations that are applicable only for this Component |

Refer to **PSoC Creator Help** > **Building a PSoC Creator Project** > **Generated Files (PSoC 6)** for information on MISRA compliance and deviations of the files generated by PSoC Creator.

The VDAC12_PDL Component has no specific deviations.

# Resources

The VDAC12_PDL Component uses the following device resources:

| Resource Type | Used |
|---|---|
| Continuous Time DAC | 1 |
| Opamp | Up to 2 |
| Sample and Hold | Up to 1 |

# DC and AC Electrical Characteristics

**Note** Final characterization data for PSoC 6 devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.

# Component Changes

This section lists the changes in the VDAC12_PDL Component from the previous versions.

| Version | Description of Changes | Reason for Changes / Impact |
|---|---|---|
| 1.0.c | Minor datasheet edits | |
| 1.0.b | Production qualified, remove "Prototype" designation | |
| 1.0.a | 1. Add an "unbuffered internal" option to the Output Buffer selections<br>2. Remove read only power consumption and gain bandwidth displays in the Advanced tab.<br>3. Rename Component to VDAC12_PDL | 1. To support an unbuffered internal connection between the VDAC12_PDL and other Components.<br>2. Displayed values may become out-of-date. Customer should refer to the device datasheet for performance specs.<br>3. To be consistent with naming of other PDL-based PSoC 6 components. Breaks backwards compatiblity with Beta release. |
| 1.0 | Initial version | |

© Cypress Semiconductor Corporation, 2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.