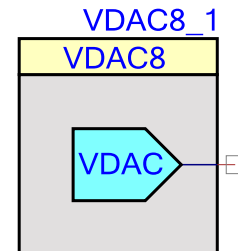


8-Bit Voltage Digital to Analog Converter (VDAC8)

1.10

Features

- Two ranges: 1.020V and 4.080V full scale
- Software or clock driven output strobe
- Data source may be CPU, DMA, or UDB
- Voltage output



General Description

The VDAC8 component is an 8-bit voltage output Digital to Analog Converter (DAC). It may be configured in a number of ways depending on your needs. The VDAC8 may be controlled by hardware, software or with a combination of both hardware and software.

When to use a VDAC8

Use the VDAC8 when you need a fixed or programmable voltage source.

Input/Output Connections

This section describes the various input and output connections for the VDAC8. An asterisk (*) in the list of I/O's states that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

Vout – Analog

The Vout terminal is the connection to the DAC's voltage output. It may be routed to any analog compatible pin on the PSoC.

Data[7:0] – Input *

This 8-bit wide data signal connects the VDAC8 directly to the DAC Bus. The DAC Bus may be driven by UDB based components, control registers, or routed directly from GPIO pins. This input is enabled by setting the "Data_Source" parameter to "DAC_Bus". If the "DMA_or_CPU" option is selected instead, the bus connection will disappear from the component symbol.

Use the Data[7:0] input when hardware is capable of setting the proper value without CPU intervention. When using this option, the Strobe option should be enabled as well.

PRELIMINARY

For many applications this input is not required, but instead the CPU or DMA will write a value directly to the data register. In firmware, use the SetRange() function or directly write a value to the VDAC8_1_Data register (assuming an instance name of “VDAC8_1”).

Strobe – Input *

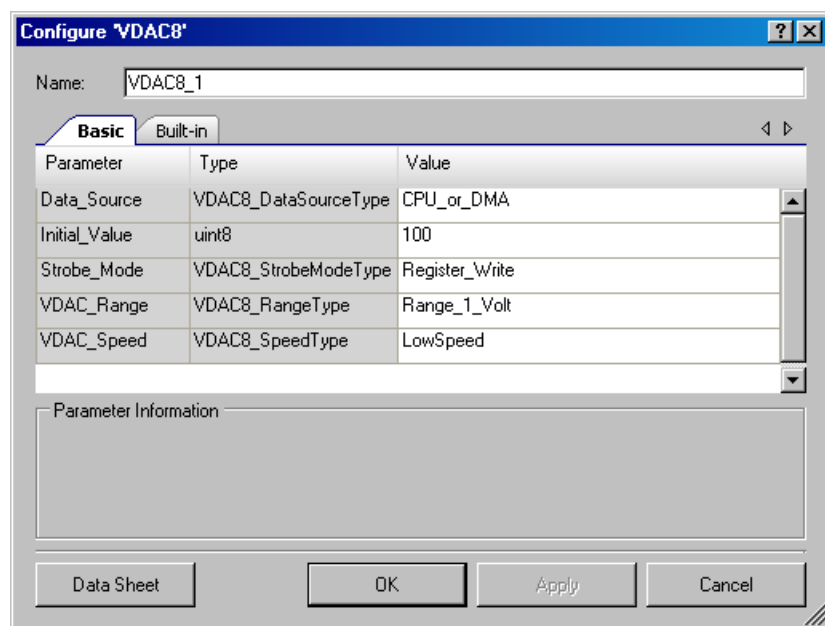
The Strobe input is an optional signal input and is selected with the “Strobe_Mode” parameter. If Strobe_Mode is set to “External”, the pin will be visible and must be connected to a valid digital source. In this mode the data is transferred from the VDAC8 register to the DAC on the next positive edge of the Strobe signal. If this parameter is set to “Register_Write” the pin will disappear from the symbol and any write to the data registers will be immediately transferred to the DAC.

For audio or periodic sampling applications, the same clock used to clock the data into the DAC could also be used to generate an interrupt. Each rising edge of the clock would transfer data to the DAC and cause an interrupt to get the next value loaded into the DAC register.

Parameters and Setup

Drag a VDAC8 component onto your design and double-click it to open the Configure dialog.

Figure 1 Configure VDAC8 Dialog



The VDAC8 component provides the following parameters.

PRELIMINARY



Data_Source

This parameter selects the source of the data to be written into the DAC register. If the CPU (firmware) or the DMA will write data to the VDAC8, then select “DMA_or_CPU”. If data is written directly from the UDBs or UDB based component, then the “DAC_BUS” should be selected. There is only one DAC Bus

Initial_Value

This is the initial value the VDAC8 will present after the Start() command is executed. The SetValue() function or a direct write to the DAC register will override the default value at anytime. Legal values are between 0 and 255 inclusive.

Strobe_Mode

This parameter selects whether the data is immediately written to the DAC as soon as the data is written into the VDAC8 data register. This mode is selected when the “Register_Write” option is selected. When the “External” option is selected, a clock or signal from the UDBs controls when the data is written from the DAC register to the actual DAC.

VDAC_Range

This parameter allows you to set one of two voltage ranges as the default value. The range may be changed at any time during runtime with the SetRange() function.

| Range | Lowest Value | Highest Value | Step Size |
|--------------|--------------|---------------|-----------|
| Range_1_Volt | 0.0 mV | 1.020 V | 4 mV |
| Range_4_Volt | 0.0 mV | 4.080 V | 16 mV |

Output equations:

- 1 Volt range – $V_{out} = (value/256) * 1.024 \text{ Volts}$
- 4 Volt range – $V_{out} = (value/256) * 4.096 \text{ Volts}$

Note “value” is a number between 0 and 255.

VDAC_Speed

This parameter provides two settings: LowSpeed and HighSpeed. In the LowSpeed mode, the settling time is slower but consumes less operating current. In the HighSpeed mode, the voltage settles much faster, but at a cost of more operating current.



PRELIMINARY

Resources

The VDAC8 uses one viDAC8 analog block.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "VDAC8_1" to the first instance of a component in a given design. You can rename the instance 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 table is "VDAC8".

| Function | Description |
|----------------------------------|---|
| void VDAC8_Start(void) | Configures the VDAC8 with the default parameters with the give power. |
| void VDAC8_Stop(void) | Disables the VDAC8 and sets it to the lowest power state. |
| void VDAC8_SetSpeed(uint8 speed) | Set DAC speed. |
| void VDAC8_SetValue(uint8 value) | Sets value between 0 and 255 with the given range. |
| void VDAC8_SetRange(uint8 value) | Sets range to 1 or 4 volts. |

void VDAC8_Start(void)

| | |
|----------------------|--|
| Description: | The start function initializes the VDAC8 with the default values, and sets the power to the given level. A power level of 0, is the same as executing the stop function. |
| Parameters: | None |
| Return Value: | None |
| Side Effects: | None |

void VDAC8_Stop(void)

| | |
|----------------------|--|
| Description: | Powers down VDAC8 to lowest power state and disables output. |
| Parameters: | None |
| Return Value: | None |
| Side Effects: | None |

PRELIMINARY



void VDAC8_SetSpeed(uint8 speed)**Description:** Set DAC speed.**Parameters:** (uint8) speed: Sets DAC speed, see table below for valid parameters.

| Option | Description |
|-----------------|-------------------------|
| VDAC8_LOWSPEED | Low speed (low power) |
| VDAC8_HIGHSPEED | High speed (high power) |

Return Value: None**Side Effects:** None**void VDAC8_SetRange(uint8 range)****Description:** Sets range to 1 or 4 volts.**Parameters:** (uint8) range: Sets full scale range for VDAC8. See table below for ranges.

| Option | Description |
|----------------|---------------------------------|
| VDAC8_RANGE_1V | Set full scale range of 1.020 V |
| VDAC8_RANGE_4V | Set full scale range of 4.080 V |

Return Value: None**Side Effects:** None**void VDAC8_SetValue(uint8 value)****Description:** Sets value to output on VDAC8. Valid values are between 0 and 255.**Parameters:** (uint8) value: Value between 0 and 255. A value of 0 is the lowest (zero) and a value of 255 is the full scale value. The full scale value is dependent on the range which is selectable with the SetRange API.**Return Value:** None**Side Effects:** None**PRELIMINARY**

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the VDAC8 component. This example assumes the component has been placed in a design with the default name "VDAC8_1."

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>

void main()
{
    VDAC8_1_Start();           // Enable VDAC8
    VDAC8_1_SetRange(VDAC8_1_RANGE_1V); // Set full scale range to 1.024V
    VDAC8_1_SetValue(100);     // Set value to 400 mV
}
```

Registers

The functions provided support most of the common runtime functions required for most applications. The following register references provide brief descriptions for the advanced user. The VDAC8_Data register may be used to write data directly to the DAC without using the API. This may be useful for either the CPU or DMA.

Table 1 VDAC8_CR0

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|------|---|---|------|------------|---|----|------|
| Value | RSVD | | | mode | Range[1:0] | | hs | RSVD |

- mode: Sets DAC to either voltage or current mode.
- range[1:0]: DAC range settings.
- hs: Use to set data speed.

Table 2 VDAC8_CR1

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|------|---|---------|---------------|---------|---------|---------|---------|
| Value | RSVD | | mx_data | reset_u db_en | mx_idir | idirbit | Mx_ioff | ioffbit |

- mx_data: Select data source.
- reset_u db_en: DAC reset enable.
- mx_idir: Mux selection for DAC current direction control.
- idirbit: Register source for DAC current direction.

PRELIMINARY



- mx_off: Mux selection for DAC current off control.
- ioffbit: Register source for DAC current off

Table 3 VDAC8_DATA

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|-----------|---|---|---|---|---|---|---|
| Value | Data[7:0] | | | | | | | |

- Data[7:0]: DAC data register.

DC and AC Electrical Characteristics

5.0V/3.3V DC and AC Electrical Characteristics

| Parameter | Typical | Min | Max | Units | Conditions and Notes |
|----------------------------------|---------|-----|-----|--------|----------------------|
| Resolution | | 8 | 8 | bits | |
| Linearity | | | | | |
| DNL | --- | | --- | LSB | |
| INL | --- | | --- | LSB | |
| Monotonic | YES | | | | |
| V _{OS} , Offset Voltage | | | | uA | |
| Output Noise | | | | uA rms | X to y MHz |
| fclock | | | | | |
| Fast mode | | | | MHz | |
| Slow mode | | | | MHz | |
| Operating Current | | | | | |
| Fast mode | | | | uA | |
| Slow mode | | | | uA | |
| | | | | | |



PRELIMINARY

© Cypress Semiconductor Corporation, 2009. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® Creator™, Programmable System-on-Chip™, and PSoC Express™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

PRELIMINARY

