6-Bit Voltage Output DAC Datasheet DAC6 V 4.3

Copyright © 2001-2015 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	flash	RAM	
CY8C29/27/24/23/22xxx, CY8CLED04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28x43, CY8C28x52						
	0	0	1	61	0	1

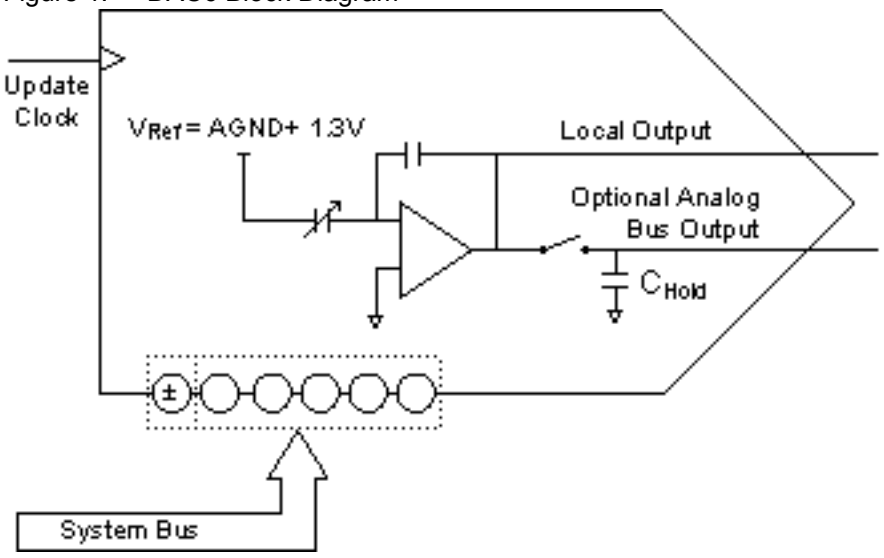
For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects.

Features and Overview

- 6-bit resolution
- Voltage output
- 2's complement, offset binary and sign/magnitude input data formats
- Sample and hold for analog bus and external outputs
- Update rates 250 ksps

The DAC6 User Module translates digital codes to output voltages. The DAC6 translates digital codes to output voltages at an update rate of up to 250k samples per second. The Application Programming Interface (API) supports offset-binary, sign-and-magnitude, and 2's complement data formats for maximum flexibility. Offset compensation is employed to minimize the error.

Figure 1. DAC6 Block Diagram

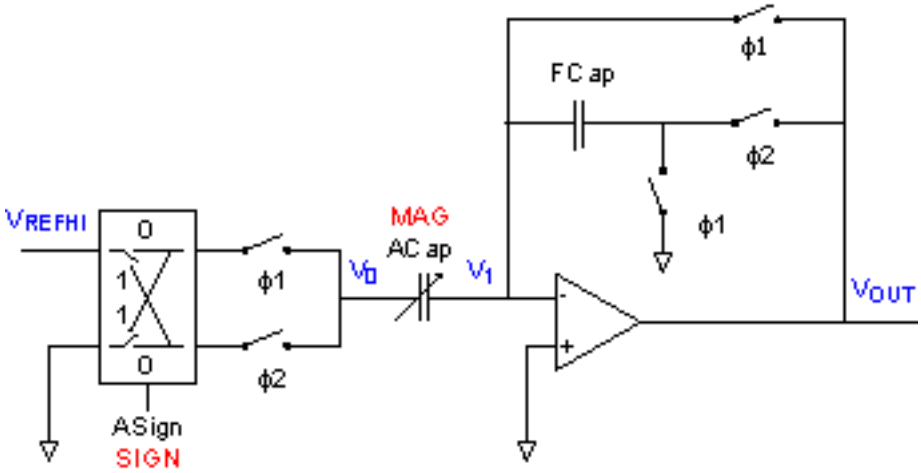


Functional Description

The DAC6 User Module converts digital codes into analog output voltages. The digital codes are represented as numbers in 2's complement or sign-and-magnitude form ranging from -31 to +31. Alternatively, input codes may be represented in offset-binary form as a number ranging from 0 to 62. This means that a one-step change in the output voltage represents one sixty-third of the full scale output range, rather than the more typical one sixty-fourth. In sign-and-magnitude form, the input code '-0' is translated to '+0' by the user module API. Several output voltage ranges are possible, depending on the value selected for a system-level parameter, REFMux.

Internally, the operation is based on sign-and-magnitude format. Five magnitude bits set the value of ACap, an array of binary-weighted capacitors shown in the simplified schematic below. ACap assumes values from 0 to 31 units. The reference voltage, which may be inverted by the ASign bit, is scaled at the output by the ratio of ACap to the feedback capacitor, FCap, with a nominal capacitance of 32 units.

Figure 2. Simplified Schematic of the DAC6



The sign bit controls the polarity, by means of a crossbar switch, in conjunction with switches controlled by the clock signals Φ_1 and Φ_2. They are equal in period and opposite in phase. Φ_1 and Φ_2 are "under-lapped". That is, there is a short period between each pulse during which neither one is active. This guarantees that Φ_1 and Φ_2 open and close their respective switches in break-before-make fashion. A clock generator, in each analog column, synchronizes the source clock to the 24 MHz oscillator and divides-by-four to produce them.

When the input code is positive (the SIGN bit is zero), the crossbar connects V_{REFHI} to ACap while Φ_1 is active, charging ACap to the reference voltage minus AGND. After Φ_1 goes inactive and as Φ_2 becomes active, the input side of ACap is switched from V_{REFHI} to AGND, effectively inverting its sense with respect to AGND. As charge is shared between ACap and FCap, the opamp supplies an opposite charge.

Thus, for a positive input code, two voltage inversions occur. The first inversion happens when the source terminal of ACap switches to AGND. The opamp itself forces a second inversion because ACap connects to its inverting input. The analysis for negative input codes is similar. The chief difference is that V_{REFHI} is applied directly during Φ_2, rather than in Φ_1. This is done so the effective inversion of the ACap charge does not occur and the only inversion is provided by the opamp.

The hardware performs offset compensation in each update cycle. Switches controlled by Φ_1 and Φ_2 configure the opamp, as a unity-gain follower during Φ_1. In this configuration, the offset voltage appears at the summing node, charging both ACap and FCap. As reconfigured in Φ_2, the circuit inverts the offset charges on these capacitors, effectively canceling the offset voltage.

On every update cycle, V_{out} slews between the opamp offset voltage (during Φ_1) and the desired voltage (settled during Φ_2); a direct result of offset compensation. One way to mitigate this price for increased accuracy is by employing the sample-and-hold circuit associated with the output bus. Vout charges both the load and the hold capacitor (CHold in the DAC6 block diagram), during the last half of Φ_2. CHold is isolated from the opamp output at the end of that period. Each analog output bus is served by an analog output buffer with suitably high input impedance.

Equation 1

$$V_{Out} = (V_{REFHI} - AGND)\frac{ACap}{FCap} + AGND = 1.3V\left(\frac{MAG}{32}\right) + 2.6V, 0 \le MAG \le 31$$

Given the REFMux parameter configured by the Device Editor to 2*BandGap ± BandGap, the following equation applies.

Example

For a value of 16 specified for the DAC input code the resultant output voltage can be expected to be 3.25V as shown in Equation 2:

Equation 2

$$V_{Out} = 1.3Volts\left(\frac{16}{32}\right) + 2.6Volts = 3.25Volts$$

The value calculated is an ideal value and will most likely differ based on system noise and chip offsets.

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the table below, TA = 25°C and Vdd = 5V. Also, fclock = 125 kHz, external AGND 2.50V, external VRef 1.23V, REFPWR = HIGH, SCPOWER = ON, PSoC block power HIGH.

Table 1. 5.0V DAC6 DC and AC Electrical Characteristics, CY8C29/27/24/22xxx Family of PSoC Devices

Parameter	Typical	Limit	Units	Conditions and Notes
Resolution	--	6	Bits	
Linearity				
DNL	0.09	--	LSB	
INL	0.07	--	LSB	
Monotonic	YES	--		
Gain Error				
Including Reference Gain Error	3.4	--	%FSR	
Excluding Reference Gain Error[3]	0.45	--	%FSR	
V_{OS}, Offset Voltage	±7.5	--	mV	
Output Noise	4.6	--	mV rms	0 to 300 kHz

Parameter	Typical	Limit	Units	Conditions and Notes
f_{clock}, Analog Column Clock[1]				
Low Power	0.128 to 0.5	--	MHz	
Med Power	0.128 to 2.0	--	MHz	
High Power	0.128 to 3.2	--	MHz	
Operating Current[2]				
Low Power	155	--	µA	
Med Power	585	--	µA	
High Power	2225	--	µA	

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the table below, TA = 25°C and Vdd = 3.3V. Also, fclock = 125 kHz, external AGND 1.50V, external VRef 0.8V, REFPWR = HIGH, SCPOWER = ON, PSoC block power HIGH.

Table 2. 3.3V DAC6 DC and AC Electrical Characteristics, CY8C29/27/24/22xxx Family of PSoC Devices

Parameter	Typical	Limit	Units	Conditions and Notes
Resolution	--	6	Bits	
Linearity				
DNL	0.09	--	LSB	
INL	0.07	--	LSB	
Monotonic	YES	--		
Gain Error				
Including Reference Gain Error	2.9	--	%FSR	
Excluding Reference Gain Error[3]	0.3	--	%FSR	
V_{OS}, Offset Voltage	±7.5	--	mV	
Output Noise	2.1	--	mV rms	0 to 300 kHz
f_{clock}, Analog Column Clock[1]				
Low Power	0.128 to 0.5	--	MHz	
Med Power	0.128 to 2.0	--	MHz	
High Power	0.128 to 3.2	--	MHz	

Parameter	Typical	Limit	Units	Conditions and Notes
Operating Current[2]				
Low Power	150	--	µA	
Med Power	560	--	µA	
High Power	2150	--	µA	

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the table below, TA = 25°C and Vdd = 2.7V. Also, fclock = 125 kHz, external AGND 1.50V, external VRef 0.8V, REFPWR = HIGH, SCPOWER = ON, PSoC block power HIGH.

Table 3. 2.7V DAC6 DC and AC Electrical Characteristics, CY8C29/27/24/22xxx Family of PSoC Devices

Parameter	Typical	Limit	Units	Conditions and Notes
Resolution	--	6	Bits	
Linearity				
DNL	0.09	--	LSB	
INL	0.07	--	LSB	
Monotonic	YES	--		
Gain Error				
Including Reference Gain Error	2.9	--	%FSR	
Excluding Reference Gain Error[3]	0.3	--	%FSR	
V_{OS}, Offset Voltage	±7.5	--	mV	
Output Noise	2.1	--	mV rms	0 to 300 kHz
f_{clock}, Analog Column Clock[1]				
Low Power	0.128 to 0.5	--	MHz	
Med Power	0.128 to 2.0	--	MHz	
High Power	0.128 to 3.2	--	MHz	
Operating Current[2]				
Low Power	150	--	µA	
Med Power	560	--	µA	
High Power	2150	--	µA	

Electrical Characteristics Notes

1. Upper end of range specified for 3dB increase in broadband noise. Lower end for droop < 1 LSB. The analog column clock selected by the DAC is 4 times faster than the phase clock rate that governs the update cycle. See the discussion of timing, below.
2. Does not include reference block power, common to all analog blocks (see the PSoC Family datasheet).
3. Reference Gain Error measured by comparing the external reference to $V_{RefHigh}$ and V_{RefLow} routed through the test mux and back out to a pin.
4. Unless otherwise specified in the table below, all limits guaranteed for $T_A = 25^{\circ}C$ and Vdd = 5V. Also, f_{clock} = 125 kHz, external AGND 2.50V, external V_{Ref} 1.23V, REFPWR = HIGH, SCPOWER = ON, PSoC block power HIGH.

Placement

The DAC6 block maps freely onto any of the switched capacitor PSoC blocks in the device. However, if the DAC6 output is enabled onto the analog output bus, care must be exercised to ensure that no other user module tries to drive the same bus. An additional consideration, in selecting a placement location, is that the clock used by the DAC6 block must also be compatible with other user module blocks mapped into the same column of PSoC blocks.

Parameters and Resources

To create a DAC6 instance, select the user module in the Device Editor, rename it if desired, and map it onto any of switched capacitor PSoC block in the device. Placement considerations include availability of an analog column output bus if the signal is to be driven off chip and interdependence with other user modules on the column clock resource. Once placed, the user module symbolically names this block “DAC" and displays its parameters.

DataFormat

The DAC6 User Module API handles three different data formats: offset binary, 2’s complement, and sign-and-magnitude. The WriteBlind entry point of the API section (below), describes these conventions and the range of values associated with each.

AnalogBus

The DAC block broadcasts its output to adjacent analog PSoC blocks. Choosing one of the analog bus options connects the DAC output to the outside world through one of the analog output buffers. In certain columns, selecting the bus provides an additional local connection to the PSoC block at the top of the array. Switched capacitor PSoC blocks incorporate a sample and hold circuit that samples the DAC output in the last half of Φ_2. This isolates external outputs from the voltage swings that occur during auto-zero operation.

ClockPhase

This parameter determines the role of the phase clocks, Φ_1 and Φ_2, generated by the column clock divider discussed in the clock and timing sections that follow. When *Normal* is selected, the auto-zero cycle occurs during Φ_1 and the output of the DAC is valid in Φ_2. When ClockPhase is set to *Swapped*, these roles are reversed. This can be useful when the DAC is connected to another peripheral that samples its input on Φ_1.

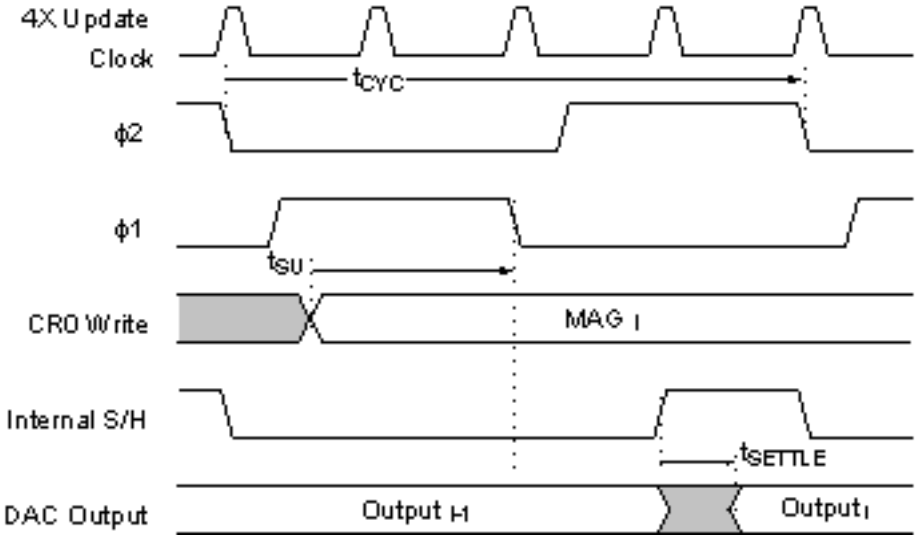
Note Setting ClockPhase to Swapped disables the sample and hold function of the analog output bus. If the AnalogBus parameter is set to Enabled, the bus output mirrors the local PSoC block output, alternating between AGND (plus the offset voltage) during Φ_1 and the desired output during Φ_2.

Analog Column Clock

The DAC continuously updates its output whether or not it is commanded to “write" an new value by calling the appropriate functions WriteBlind and WriteStall API functions. The analog column clock multiplexors selects the source clock used to generate the phase clocks, Φ_1 and Φ_2 that control this update operation. The phase clock generator divides the column clock by four to produce Φ_1 and Φ_2, so the column clock frequency is four times faster than the actual analog output update rate. Two levels of multiplexing provide choices for the column clock that include any of the digital blocks and the system clock dividers. The Electrical Characteristics section, above, specifies lower and upper limits for the column clock frequency.

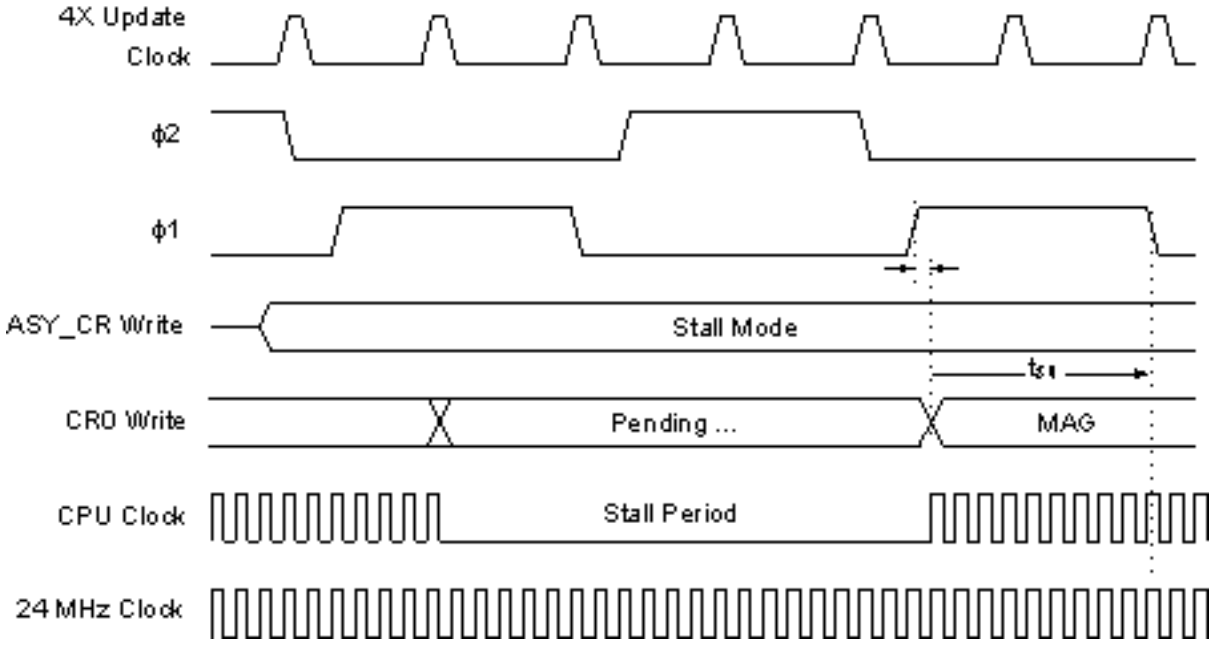
For positive output voltages (Vout > AGND) and normal phase, the reference voltage is stored on the A Cap during Φ_1 as shown in the simplified schematic, above. In order to fully charge the A Cap, any change to it’s value must meet set-up time t_{SU} to the falling edge of Φ_1. This set-up time, illustrated in the figure below, is dependent on the reference power levels. Although the set-up time is not characterized, the hardware stall mechanism can be used to guarantee that it will be met. If the A Cap does not fully charge due to set-up time violations the output will be incorrect until the next phase clock cycle when the entire period of f1 when it will be corrected. A similar set-up time relative to the falling edge of f2 governs behavior for negative output voltages (Vout < AGND).

Figure 3. Update Timing for V_{OUT} > AGND



For a large class of applications, momentary (one update-cycle) deviations are acceptable. Other application may impose stricter requirements. Hardware synchronization may be employed to control the timing of the register write that changes the value of the A Cap. This is directly supported in the WriteStall API by entry point. When invoked, the underlying hardware recognizes the write to the PSoC block register and freezes the CPU clock, holding off completion the write until the rising edge of Φ_1. The ASY_CR register controls.

Figure 4. Hardware Synchronization and CPU Clock Stall



During the CPU stall, all analog and digital PSoC blocks function normally. The MOV instruction that writes the DAC's CR0 register is simply suspended and, during this period, any interrupts become or remain pending. The number of CPU cycles lost during the stall equal, in time, the period of the can be calculated using the following relation.

Equation 3

$$\text{CPU Cycles} \leq \frac{F_{CPU}}{F_{\phi_1}} = \frac{4 \times F_{CPU}}{F_{ColClock}}$$

Clearly, to minimize the CPU cycles lost to the stall, the column clock should be run at the highest practical frequency. This can be much higher than necessary for actual changes to the output voltage as extra output cycles simply repeat the previous output cycle. A faster column clock also minimizes the latency from calling the output function to the time the output changes.

There are practical limits to the column clock frequency, however. Since the DAC must slew from AGND to the output voltage each phase-clock cycle, the column clock is limited to frequencies that permit the output to settle. When the sample and hold feature of the analog output bus is used, the opamp output drives the bus during the sampling window in the second half of $\Phi 2$. If the column clock is so fast that the opamp is still slewing to and settling on the output voltage, this becomes observable as noise on the output signal. In extreme cases, the output will slew only part way to the final output voltage before the sampling window closes. This may be observed as severe non-linear gain compression on the output most evident at the full-scale ends of the output range. Upper limits for the

analog column clock that prevent this from occurring are provided in the Electrical Characteristics tables, above.

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the “include" files.

Note

In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X prior to the call if those values are required after the call. This “registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API function may leave A and X unchanged, there is no guarantee they will do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

Entry points are provided to initialize the DAC6 User Module, write updated values, and disable the user module.

DAC6_Start

Description:

Performs all required initialization for this user module and sets the power level for the switched capacitor PSoC block.

C Prototype:

```
void  DAC6_Start(BYTE bPowerSetting)
```

Assembler:

```
mov   A, bPowerSetting
lcall  DAC6_Start
```

Parameters:

bPowerSetting: One byte that specifies the power level. Following reset and configuration, the PSoC block assigned to the DAC block is powered down. Symbolic names, provided in C and assembly, and their associated values, are given in the following table.

Symbolic Name	Value
DAC6_OFF	0
DAC6_LOWPOWER	1
DAC6_MEDPOWER	2
DAC6_FULLPOWER	3

Return Value:

None

Side Effects:

The DAC outputs will be driven. By default, the initial value is AGND. Call one of the Write routines prior to calling “Start," if some other output value is required at power on. The A and X registers may be altered by this function.

DAC6_SetPower

Description:

Sets the power level for the DAC switched capacitor PSoC block. May be used to turn the block Off and On.

C Prototype:

```
void  DAC6_SetPower(BYTE bPowerSetting)
```

Assembler:

```
mov   A, bPowerSetting
lcall  DAC6_SetPower
```

Parameters:

bPowerSetting: Identical to the PowerSetting parameter used for the Start entry point.

Return Values:

None

Side Effects:

The DAC outputs will be driven. By default, the initial value is AGND. Call one of the Write routines prior to calling “Start," if some other output value is required at power on. The A and X registers may be altered by this function.

DAC6_WriteBlind

Description:

Immediately updates the output voltage to the indicated value.

C Prototypes:

```
void  DAC6_WriteBlind(CHAR cOutputValue)
```

Assembler:

```
mov   A, cOutputValue
lcall  DAC6_WriteBlind
```

Parameters:

cOutputValue: One byte that specifies the output voltage. Allowed values lie in the range corresponding to the selected value of DataFormat, as given in the following table.

Data Format	Minimum	Maximum
OffsetBinary	0	62
TwosComplement	-31	31
SignAndMagnitude	-31	31

TwosComplement and OffsetBinary use the native 2's complement format of the microcontroller. Offset-binary values are positive numbers with the lowest output voltage represented by zero and the highest by 62. In SignAndMagnitude format, the byte is required to have the binary form "00smmmmm," where mmmmm is the magnitude and s is the sign. Encode s using 0 for positive numbers and 1 for negative numbers.

Return Values:

None

Side Effects:

The output may glitch for reasons discussed in the Timing section in this user module. The A and X registers may be altered by this function.

Note: When you select the OFFSET_BINARY, input values that are out of range are automatically converted to two's compliment data within the API. This means that an out of range value, above the maximum allowed offset binary value, is converted to a small positive output (near Agnd).

DAC6_WriteStall

Description:

Possibly stalls the microprocessor until the beginning of Φ_1, then updates the output voltage to the indicated value. Note that the API assumes that either interrupts are disabled or the maximum interrupt latency is less than ACLKi.

C Prototypes:

```
void  DAC6_WriteStall (CHAR cOutputValue)
```

Assembler:

```
mov   A, cOutputValue
lcall  DAC6_WriteStall
```

Parameters:

cOutputValue: Identical in format and value range to the cOutputValue parameter described for the WriteBlind entry point.

Return Values:

None

Side Effects:

If ACLKi is inactive (where 'i' is the column into which the analog PSoC block is mapped), the microprocessor's CPU clock is disabled until Φ_2 goes inactive, possibly for three-quarters of an update cycle (plus two CPU clocks). Note that no interrupts are recognized during the stall interval. The A and X registers may be altered by this function.

DAC6_Stop

Description:

Powers the user module Off.

C Prototype:

```
void  DAC6_Stop(void)
```

Assembly:

```
lcall  DAC6_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

Outputs will not be driven. The A and X registers may be altered by this function.

Sample Firmware Source Code

The following sample code creates a periodic, descending sawtooth wave:

```
//-------------------------------------------------------------------------
// This C sample code for the DAC6 User Module creates a periodic signal
// that ramps down.
//-------------------------------------------------------------------------

#include <m8c.h>            // part specific constants and macros
#include "PSoCAPI.h"        // PSoC API definitions for all user modules


#define DAC_MAX (62)                    // Define max DAC value as 62

unsigned char bDACValue = 0;            // Variable for the DAC value
unsigned char i;                        // Variable for an index

void main(void)
{
    DAC6_Start(DAC6_HIGHPOWER);         // Start DAC6 in HIGH power mode

while(1)                            // Repeat forever
    {
        if(bDACValue == 0)
    {
           bDACValue = DAC_MAX;        // Reset DAC value to the max if it reached zero
        }

    DAC6_WriteStall(bDACValue--);   // Write value to DAC and decrement

    for(i = 0xFF; i != 0; i--);     // Delay loop
}
}
```

The following assembly sample code has the same functionality as the C sample code:

```
;-------------------------------------------------------------------------
; This sample code for the DAC6 User Module generates a periodic signal that
; ramps down
;-------------------------------------------------------------------------

include "m8c.inc"           ; part specific constants and macros
include "memory.inc"        ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"       ; PSoC API definitions for all user modules

export _main

DAC_MAX: equ 62             ; This is the maximum DAC value

area bss (RAM, REL, CON)
    bDACValue: blk 1        ; Variable to hold the DAC value


area text (ROM, REL, CON)
_main:
mov A, DAC6_HIGHPOWER    ; Start DAC with HIGH power setting
    call DAC6_Start

Init:
    mov [bDACValue], DAC_MAX ; Initialize DAC value to hold the maximum
    mov X, 0xFF              ; Initialize X register to hold 0xFF

RampDown:
    mov A, [bDACValue]      ; Move DAC value into A register
    call DAC6_WriteStall    ; Write the value in A to the DAC

Delay:
    dec X                   ; Decrement X register
    jnz Delay               ; Keep delaying if it hasn't reached zero yet

dec [bDACValue]         ; Decrement DAC value variable
    jnz RampDown            ; If it is not zero, keep ramping down
    jmp Init                ; If it is zero, restart the ramp down
```

Configuration Registers

The API provides a complete interface to the DAC6 User Module. Writing directly to the configuration registers affords an alternative means of updating the output. Either way, there are timing considerations which must be understood to prevent output glitches. The following registers are used for the DAC6 switched capacitor DAC block.

Table 4. Block DAC ASAxxCR0 or ASBxxCR0: Register CR0

Bit	7	6	5	4	3	2	1	0
Value	1	0	Sign and Magnitude					

Sign and Magnitude is set to mid-scale (AGND) following reset and reconfiguration. It is modified by way of the “Write" calls in the API.

Table 5. Block DAC ASAxxCR1 or ASBxxCR1: Register CR1

Bit	7	6	5	4	3	2	1	0
Value	0	1	0	0	0	0	0	0

Table 6. Block DAC ASAxxCR2 or ASBxxCR2: Register CR2

Bit	7	6	5	4	3	2	1	0
Value	Analog Bus	0	1	0	0	0	0	0

AnalogBus determines whether the DAC PSoC block drives the bus. The value of this bit-field is determined by the choice made, for the parameter of the same name, in the user module Placement mode of the Device Editor.

Table 7. Block DAC ASAxxCR3 or ASBxxCR3: Register CR3

Bit	7	6	5	4	3	2	1	0
Value	0	0	1	1	0	0	Power	

Power is set to Off following device reset and configuration. It is modified by calling Start, SetPower, or Stop entry points in the API.

Table 8. Global Register ASY_CR

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	0	0	0	1

The API writes to this register when required, stalling the CPU in order to guarantee output update timing requirements.

Version History

Version	Originator	Description
4.3	DHA	Added Version History

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.

Copyright © 2001-2015 Cypress Semiconductor Corporation. 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 Designer™ and Programmable System-on-Chip™ 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.