8-Bit Voltage Output Multiplying DAC Datasheet MDAC8 V 2.2

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	
CY8C29/27/26/25/24/22xxx, CY8C23x33, CY8CLED04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CPLC20, CY8CLED16P01, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C28x43, CY8C28x52						
	0	0	2	255	0	1

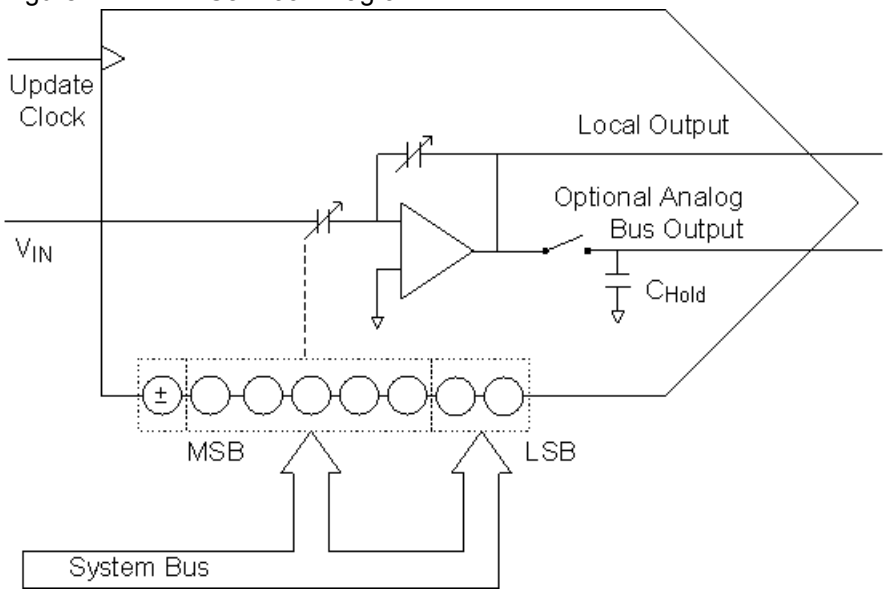
For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects.

Features and Overview

- 8-bit resolution
- Voltage output
- Four quadrant multiplication
- 2's complement, offset binary, and sign/magnitude input data formats
- Sample and hold for analog bus and external outputs
- Update rates up to 125 ksps

The MDAC8 is an 8-bit, four-quadrant multiplying DAC that scales input voltage with digital codes. The MDAC8 translates digital codes to output voltages at an update rate of up to 125k samples per second. The Application Programming Interface (API) supports offset-binary, 2's complement, and sign-and-magnitude data formats. Offset compensation minimizes conversion error.

Figure 1. MDAC8 Block Diagram

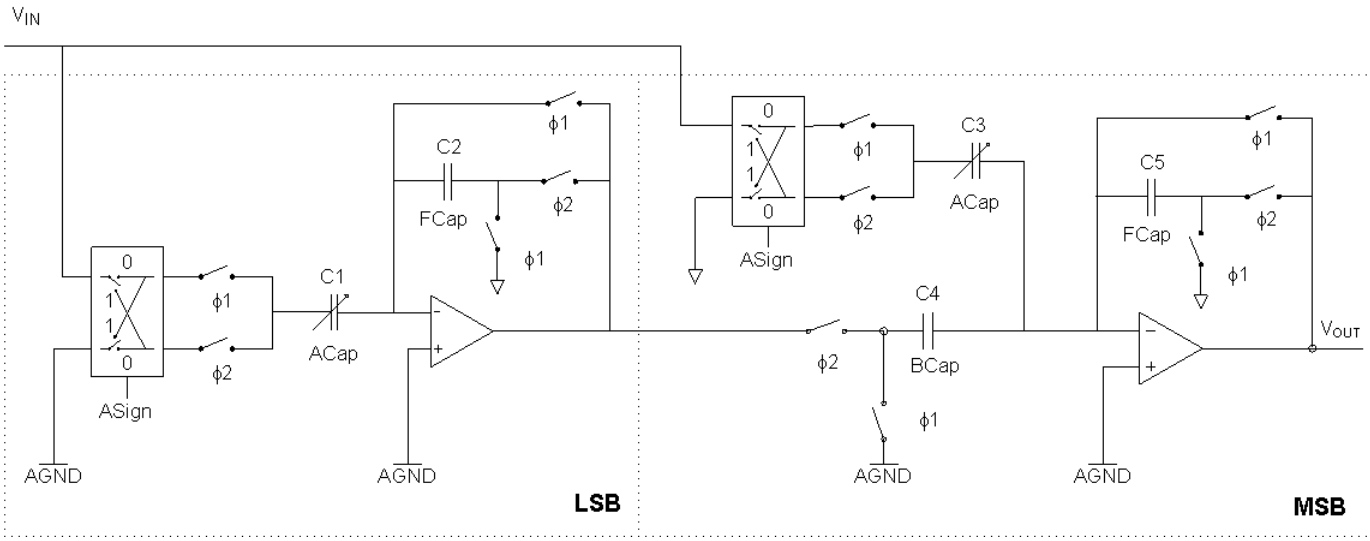


Functional Description

The MDAC8 User Module multiplies analog input voltages with digital codes. The digital codes are represented as numbers in 2's complement or sign-and-magnitude form, ranging from -127 to +127. Alternatively, input codes may be represented in offset-binary form, as a number ranging from 0 to 254. Input and output voltages are referenced to AGND, which is selected with the value in the system-level parameter, RefMux. The input voltage is multiplied either by 1 or 2, depending on the value selected for the SetOutputRange API function.

The MDAC8 User Module maps onto any two analog PSoC blocks. These blocks are designated LSB and MSB. The LSB block, or stage, is coupled to the MSB stage through a "BCap" capacitor, C_4. Internally, the operation is based on sign-and-magnitude format. The five most significant magnitude bits set the value of C_3, an array of binary-weighted capacitors shown in the simplified schematic below. The two least significant magnitude bits set the value of C_1. C_3 assumes values from 0 to 31 units and C_1 assumes values from the set {0, 8, 16, 24} in units of capacitance. The input voltage, which may be inverted by the ASign bit, is scaled in each stage by the ratio of the magnitude capacitors, C_1 and C_3, to the feedback capacitors, C_2 and C_5, respectively. Each assumes values of 32 or 16 units (32 units gains the input voltage by 1, 16 units by 2). The output of the LSB stage is further scaled by the ratio of the coupling capacitor C_4 to the C_5 feedback capacitor. The SetOutputRange API function changes the values of both C_2 and C_5.

Figure 2. Simplified Schematic of the MDAC8



The hardware performs offset compensation in each update cycle. Switches controlled by φ_1 and φ_2 configure the opamps as unity-gain followers during φ_1. In this configuration, the offset voltages appear at the summing nodes, charging the various ACaps, BCaps, and FCaps. As reconfigured in φ_2, the circuit inverts the offset charges on these capacitors, effectively canceling the offset voltages.

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. V_{out} charges both the load and the hold capacitor (C_{Hold} in the MDAC8 block diagram), during the last half of φ_2. C_{Hold} 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.

Combining the scaled inputs, the output is as follows.

Equation 1

$$V_{Out} = (V_{IN}–AGND)\frac{C_1C_4}{C_2C_5} + (V_{IN}–AGND)\frac{C_3}{C_5} + AGND$$

When the global parameter RefMux is configured to (2 BandGap) ± BandGap in the Device Editor, AGND is 2.6 volts. Set $C_F = C_2 = C_1$, $C_4 = 1$, and $C_F = 2^5$. The corresponding output is as follows.

Equation 2

$$V_{Out}= (V_{IN} - 2.6)\frac{C_1}{2^5 2^5} + (V_{IN} - 2.6)\frac{C_3}{2^3} + 2.6$$

Equation 3

$$V_{Out}= V_{IN}\left(\frac{C_1}{2^{10}} + \frac{C_3}{2^5}\right) - 2.6\left(\frac{C_1}{2^{10}} + \frac{C_3}{2^5}\right) + 2.6$$

Recall that C_1 is constrained to values obtained scaling the two least significant magnitude bits by a factor of eight (shift left three places). Canceling the scale factor in C_1 and in its denominator, the result can be expressed as follows.

Equation 4

$$V_{Out}= 2.6Volts + V_{IN}\left(\frac{C_1}{2^8} + \frac{C_3}{2^5}\right) - 2.6\left(\frac{C_1}{2^8} + \frac{C_3}{2^5}\right), C_1 \in 0, 1, 2, 3$$

Example

Given an input voltage of 1V provided by an external source then Equation 3 becomes as follows:

Equation 5

$$V_{Out} = 2.6Volts \pm 1.6Volts\left(\frac{C_1}{2^8} + \frac{C_3}{2^5}\right), C_1 \in 0, 1, 2, 3$$

The input codes to the MDAC are therefore distributed across the sign which is the MSB of the input code. The value of C_3 which is contained in Bits 2 - 7. Finally the value of C_1 which is represented by the 2 LSB bits of the input code. So for a value of -105 the output voltage would be 1.3V.

Equation 6

$$V_{Out} = 2.6Volts–1.6Volts\left(\frac{1}{256} + \frac{26}{32}\right) = 1.3V$$

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

Four-quadrant multiplication means both input voltage and input code can cause output voltage to be either positive or negative. See the following three figures.

Figure 3. Input Voltage versus Time

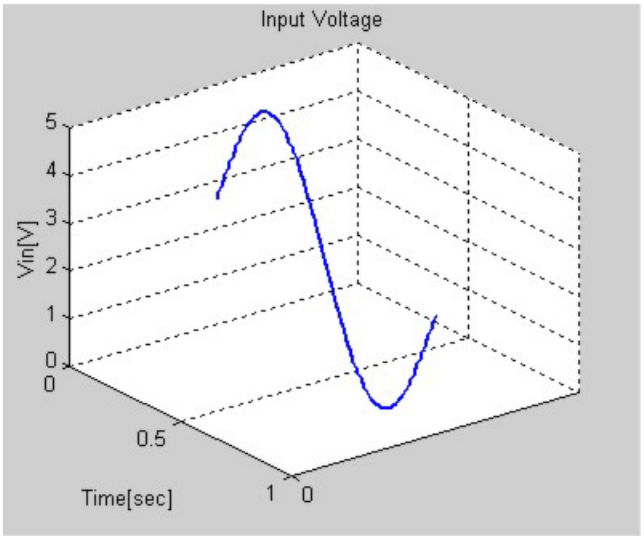


Figure 4. Output Voltage versus Input Code and Time, FCap=32

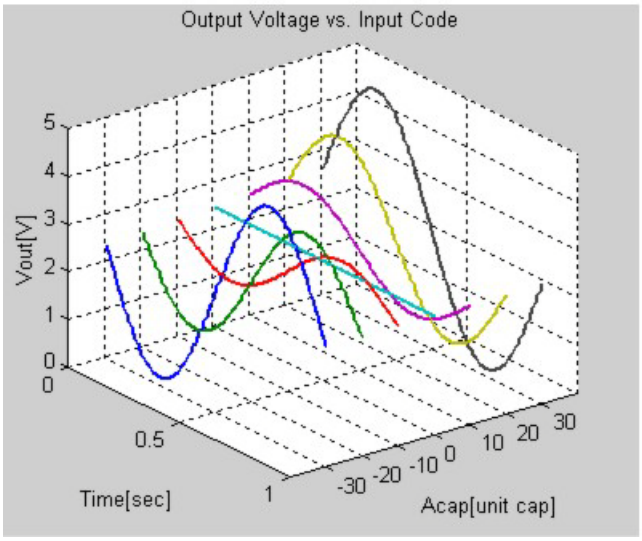
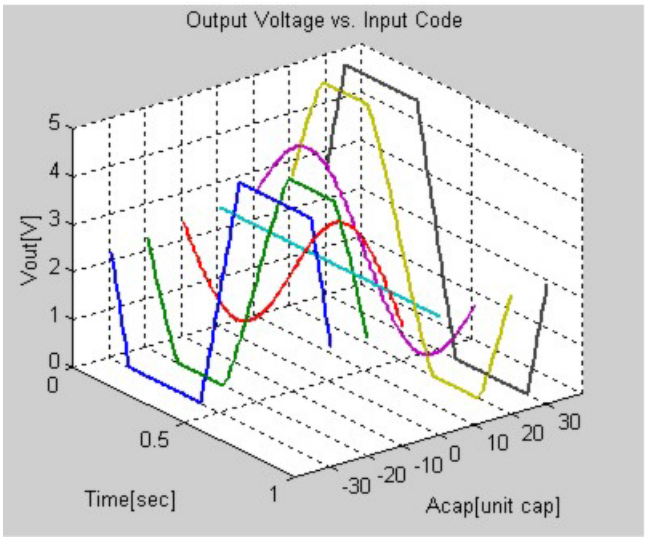


Figure 5. Output Voltage versus Input Code and Time, FCap=16



Input voltage must be decreased to keep the output from clipping, as shown above.

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 = 25oC and Vdd = 5V. Unless otherwise noted, fclock = 125 kHz, external AGND 2.50V, external VRef 1.23V, REFPWR = HIGH, SCPOWER = ON, PSoC block power HIGH.

Table 1. 5.0V MDAC8 DC and AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Resolution	--	8	Bits	
Linearity				
DNL	0.5	--	LSB	
INL	0.3	--	LSB	
Monotonic	YES	--		
Gain Error				
Including Reference Gain Error	3.5	--	%FSR	
Excluding Reference Gain Error[3]	0.5	--	%FSR	
V_{OS}, Offset Voltage	±2.1	--	mV	

Parameter	Typical	Limit	Units	Conditions and Notes
Output Noise	4.5	--	mV rms	0 to 300 kHz
f_{clock}, Internal Update Rate[1]				
Low Power	2 to 125	--	kHz	
Med Power	1 to 500	--	kHz	
High Power	1 to 800	--	kHz	
Operating Current[2]				
Low Power	305	--	µA	
Med Power	1130	--	µA	
High Power	4315	--	µA	

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

Table 2. 3.3V DAC8 DC and AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Resolution	--	8	Bits	
Linearity				
DNL	0.5	--	LSB	
INL	0.4	--	LSB	
Monotonic	YES	--		
Gain Error				
Including Reference Gain Error	2.6	--	%FSR	
Excluding Reference Gain Error[3]	0.3	--	%FSR	

Parameter	Typical	Limit	Units	Conditions and Notes
V_{OS}, Offset Voltage	±3.5	--	mV	
Output Noise	2.5	--	mV rms	0 to 300 kHz
f_{clock}, Internal Update Rate[1]				
Low Power	2 to 125	--	kHz	
Med Power	1 to 500	--	kHz	
High Power	1 to 800	--	kHz	
Operating Current[2]				
Low Power	270	--	µA	
Med Power	1020	--	µA	
High Power	3900	--	µA	

Electrical Characteristics Notes

1. Limit for φ_1, φ_2; specified for 3dB increase in broadband noise.
2. Does not include reference block power, common to all analog blocks (see the PSoC Family datasheet).

Unless otherwise specified in the table below, all limits guaranteed for TA = 25C and Vdd = 5V. Unless otherwise noted, fclock = 125 kHz, external AGND 2.50V, external VRef 1.23V, REFPWR = HIGH, SCPOWER = ON, PSoC block power HIGH.

Table 3. 5.0V MDAC8 DC and AC Electrical Characteristics

Parameter	Typical[1]	Limit[2]	Units	Conditions and Notes
Resolution	—	8	Bits	
Linearity				
DNL	0.10	0.25	LSB	
INL	0.15	0.40	LSB	

Parameter	Typical[1]	Limit[2]	Units	Conditions and Notes
Monotonicity	—	½	Bit	
Gain Error	1.0	2.5	%FSR	
V_{OS}, Offset Voltage[3]	8	43	mV	
Output Noise				
Band Limited	0.3	1	mV rms	0 to 10 kHz
Broad Band	7	10	mV rms	0 to 300 kHz
f_{clock}, Internal Update Rate[4]	—	32 to 333	kHz	
V_{in} Bandwidth	40	—	kHz	
Operating Current[5]				
Low Power	250	—	µA	
Med Power	560	—	µA	
High Power	1560	2000	µA	

Unless otherwise specified in the table below, all limits guaranteed for T_A = 25°C and Vdd = 3.3V. Unless otherwise noted, f_{clock} = 125 kHz, external AGND 1.50V, external V_{Ref} 0.80V, REFPWR = HIGH, SCPOWER = ON, PSoC block power HIGH.

Table 4. 3.3V MDAC8 DC and AC Electrical Characteristics

Parameter	Typical[1]	Limit[2]	Units	Conditions and Notes
Resolution	—	8	Bits	
Linearity				
DNL	0.10	0.20	LSB	
INL	0.20	0.45	LSB	

Parameter	Typical[1]	Limit[2]	Units	Conditions and Notes
Monotonicity	—	½	Bit	
Gain Error	1.0	2.5	%FSR	
V_{OS}, Offset Voltage[3]	7	31	mV	
Output Noise				
Band Limited	0.3	1	mV rms	0 to 10 kHz
Broad Band	7	10	mV rms	0 to 300 kHz
f_{clock}, Internal Update Rate[4]	—	32 to 333	kHz	
V_{in} Bandwidth	40	—	kHz	
Operating Current[5]				
Low Power	200	—	µA	
Med Power	500	—	µA	
High Power	1280	1800	µA	

Electrical Characteristics Notes

1. Typical values represent statistical mean plus 1σ.
2. Limits guaranteed by testing or statistical analysis.
3. 2’s complement zero scale offset to external AGND, does not include analog output buffer offset error.
4. Limit for φ_1, φ_2; specified for 3dB increase in broadband noise.
5. Does not include reference block power, common to all analog blocks (see the PSoC Family data-sheet).

Timing

In order to generate the timing signals and proper duty cycles for the under-lapped phase clocks, φ_1 and φ_2, the analog column clock circuits employ a pair of divide-by-2 circuits. Consequently, the clock source selected for the MDAC8 must run at least four times as fast as the required maximum update rate. In addition to the sample-and-hold signal used internally to de-glitch the output, a “Ready" signal is generated. Ready signifies Control Register 0 may be written without violating the setup time-requirement. See the Update Timing diagram below.

Failure to write both registers, in the same update cycle and meet the setup time requirement for both writes, will result in an incorrect output value for that cycle. If both writes occur prior to the onset of φ_2 but

the setup time is not met, the output will fall between the previous output voltage and the desired output. If one or both writes occur while φ_2 is high, the output value may lie outside the voltage interval defined by the previous and desired outputs. To minimize the deviation, which could be produced by non-deterministic events such as interrupts, write the MSB register first.

For a large class of applications, momentary (one update cycle) deviations, such as those just described, are acceptable. In others, stricter requirements are necessary. For example, a low-distortion waveform generator. Hardware synchronization, a method of timing the register updates, may be employed to avoid this and is directly supported in the API by entry points that end in the word “Stall."

Hardware synchronization guarantees the earliest possible access to the output-value registers. Writing the ASY_CR register triggers hardware synchronization. The next write to IO space proceeds immediately, if ACLKi is active; if not, the CPU clock is stalled partway through the IO write until ACLKi is asserted. One way to minimize the CPU cycles lost to the stall, is to run the update clock at higher frequencies (up to

f_{MAX}), even when the desired update rate is much slower. The Forced Synchronization with Fast Update Clock illustrates a setup time failure at Label “A," causing a worst-case stall of the CPU clock.

Figure 6. Update Timing

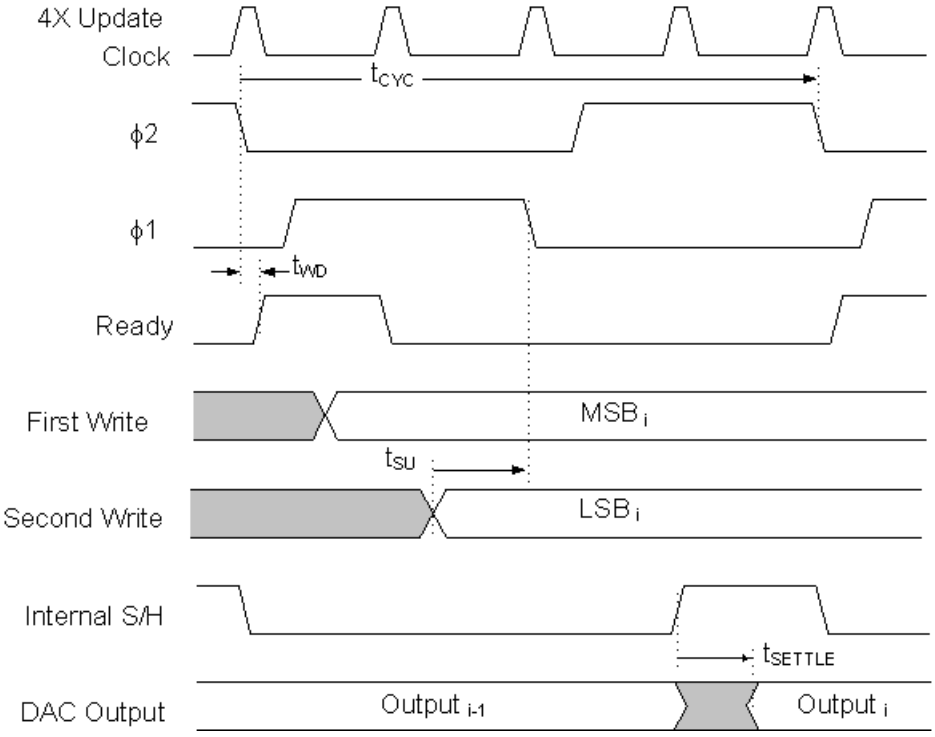
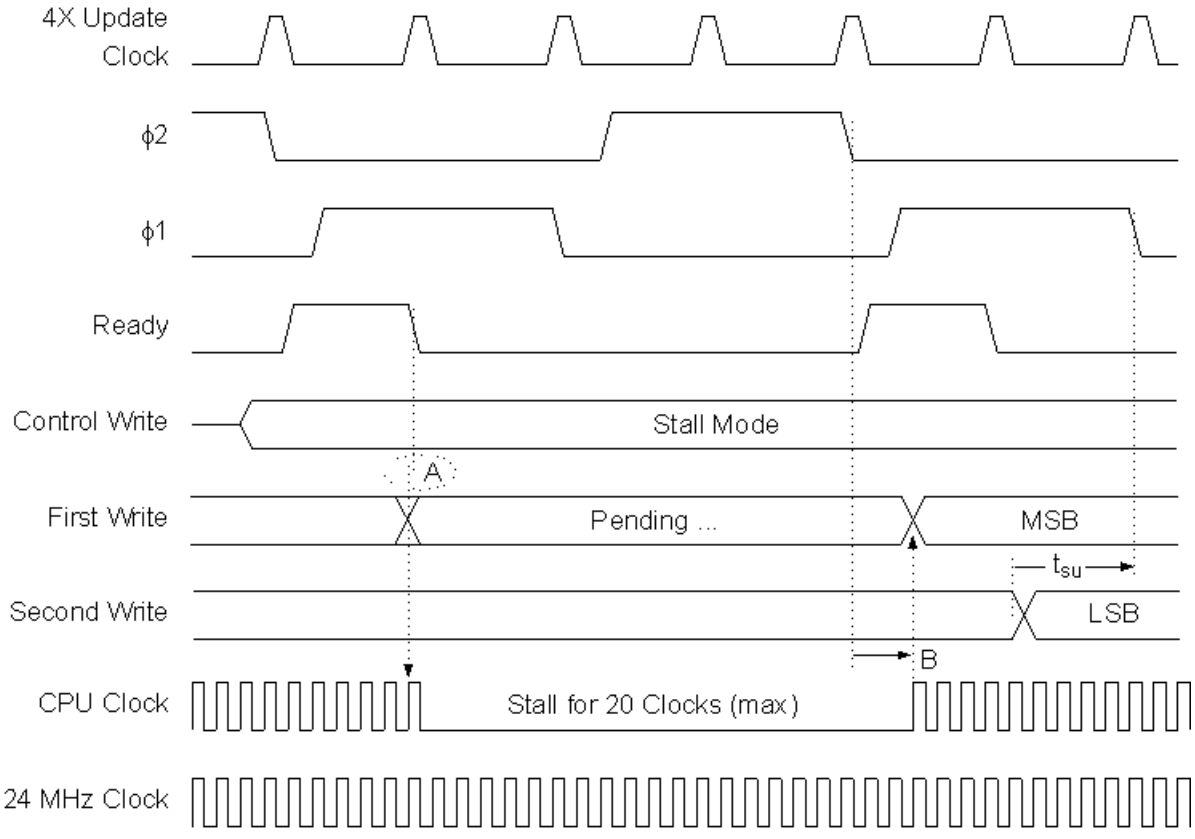


Figure 7. Forced Synchronization with Fast Update Clock



During the CPU stall, all analog and digital PSoC blocks function normally. The first, of the MOV instruction that writes the DAC CR0 register, is simply suspended and, during this period, any interrupts become or remain pending. Upon release from the stall and completion of the stalling write, pending interrupts will be processed if they are enabled. This may or may not be problematic, depending on the maximum total interrupt latency and the period of the φ_1 and φ_2 clocks.

Placement

The MDAC8 User Module maps onto two PSoC blocks, designated LSB and MSB. The output of the LSB block is fed into the input of the MSB block therefore they are always placed adjacent to each other. In the CY8C26/25xxxdevice family the MSB block maps only to “Type A" switched capacitor PSoC blocks. In the CY8C27/24/22xxxdevice family the MSB block maps only to “Type C" switched capacitor PSoC Blocks. This helps to minimize linearity errors because, these types of blocks permit the auto-zero process to cancel offset errors across the “BCap" capacitor (C_4 in the figure above) that couples the LSB and MSB blocks together.

An additional consideration, in selecting a placement location, is that the MSB and LSB clocks must be derived from the same source. This happens automatically, if they are placed in the same column in the analog array. If they are placed in two different columns, both column multiplexers must be set to the same source. Some MDAC8 placements will not allow the same input source selection for both the MSB and LSB blocks.

Note Choose the same input voltage source for both the MSB and the LSB.

Parameters and Resources

To make a working Digital-to-Analog converter, create an instance of the MDAC8 User Module in the select user module mode of the Device Editor. Next, map its MSB and LSB blocks onto the switched capacitor PSoC blocks in the analog array. Additional tasks include configuring update clock resources suitable for deriving φ_1 and φ_2, specifying the data format, choosing a GAIN range, selecting an input voltage source, and allocating the output bus associated with that PSoC block.

Configuring update clock resources involves three steps. First, configure a clock source for the analog column clock generator. The column clock generator divides its input by four to produce φ_1 and φ_2, so the source must run four times faster than the desired analog output update rate. The Timing section discusses issues pertinent to selecting an update clock frequency. Choices for the clock source include the V1 and V2 dividers, and any of the digital PSoC blocks. All of the Timer, Counter, and Pulse-Width Modulator (PWM) User Modules make suitable choices for a rate generator, when an external source is used or when V1 and V2 must be used for other purposes. The dead band time between the active periods of φ_1 and φ_2 are generated synchronously with the 24 MHz system clock. Therefore, external clock sources not synchronized to the system clock may produce unpredictable results.

Second, connect the clock source to the column clock generator by setting the CLK mux in the Device Editor. Digital PSoC block outputs must also be connected through the ACLK0 or ACLK1 multiplexers. For additional information, see the *PSoC Designer: Integrated Development Environment User Guide and the PSoC Datasheet*.

Finally, select a value for the MDAC8 User Module parameter ClockPhase, by choosing Normal (default value) or Swapped. This can synchronize the output of the MDAC8 to the input of another PSoC block. Switched capacitor analog PSoC blocks use φ_1 and φ_2 to acquire and transfer signal. Because the output of the MSB block is valid only during φ_2, a problem arises when it is connected to another user module that samples its input during φ_1. Setting the ClockPhase parameter to Swapped interchanges the roles of

φ_1 and φ_2 within both the MSB and LSB blocks, so that the output will be valid when the downstream user module samples its input. (Note that in Normal mode, the input voltage is sampled during φ_1.)

Note Choose the same input voltage source for both the MSB and the LSB.

InputMSB

Input voltage source for MSB block. Selecting REFHI, as the input voltage source, causes the MDAC8 to behave like the DAC8. Other input voltage selections require the configuration of an appropriate user module in the selected block. Some MDAC8 placements will not allow the same input source selection for both the MSB and LSB blocks.

Note Choose the same input voltage source for both the MSB and the LSB.

InputLSB

Input voltage source for LSB block. Selecting REFHI, as the input voltage source, causes the MDAC8 to behave like the DAC8. Other input voltage selections require the configuration of an appropriate user module in the selected block. Some MDAC8 placements will not allow the same input source selection for both the MSB and LSB blocks.

Note Choose the same input voltage source for both the MSB and the LSB.

AnalogBus

The MDAC8 output may be routed through the analog PSoC block array's network of local interconnections and/or through an analog output bus. Setting the MDAC8 User Module AnalogBus parameter to Disable (default value) restricts the set of possible connections to the local network. Choosing Enable extends the set of possible connections to an associated analog output buffer that can drive a pin and to some additional input multiplexers not afforded by the local network.

Each switched capacitor PSoC block incorporates a circuit that samples the bus-enabled signal on φ_2. This can eliminate the voltage swings that occur during auto-zero operation.

Note Setting AnalogBus to Enabled and ClockPhase to Swapped disables the sample and hold function. In this case, the bus output mirrors the local PSoC block output, alternating between AGND (plus the offset voltage) during φ1 and the desired output during φ2.

ClockPhase

The selection of the Clock Phase is used to synchronize the output of one analog PSoC block to the input of another. The switched capacitor analog PSoC blocks use a two-phase clock (φ_1, φ_2) to acquire and transfer signal. Setting the ClockPhase parameter to Swapped interchanges the roles of φ_1 and φ_2 within both the MSB and LSB blocks, so that the output will be valid when the downstream user module samples its input. (Note that in Normal mode, the input voltage is sampled during φ_1.)

GainRange

For a given input code, input voltage, and AGND, changing gain range from low to high will increase the output voltage appropriately. Note that input voltage range, at high gain range, effectively is half of the input voltage range, at low gain range.

DataFormat

The MDAC8 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 in this user module, describes these conventions and the range of values associated with each.

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 MDAC8 User Module, write updated values, and disable the user module.

MDAC8_Start

Description:

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

C Prototype:

```
void  MDAC8_Start(BYTE bPowerSetting)
```

Assembler:

```
mov   A, bPowerSetting
lcall  MDAC8_Start
```

Parameters:

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

Symbolic Name	Value
MDAC8_OFF	0
MDAC8_LOWPOWER	1
MDAC8_MEDPOWER	2
MDAC8_FULLPOWER	3

Return Value:

None

Side Effects:

The MDAC output 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.

MDAC8_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  MDAC8_SetPower(BYTE bPowerSetting)
```

Assembler:

```
mov   A, bPowerSetting
lcall  MDAC8_SetPower
```

Parameters:

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

Return Values:

None

Side Effects:

The MDAC outputs will be driven. By default, the initial voltage 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.

MDAC8_SetOutputRange

Description:

Sets one of two ranges for the MDAC switched capacitor PSoC block, by setting FCap to 32 (low range: gain=1) or 16 (high range: gain=2).

C Prototype:

```
void  MDAC8_SetOutputRange(BYTE bRangeSetting)
```

Assembler:

```
mov   A, bRangeSetting
lcall  MDAC8_SetOutputRange
```

Parameters:

RangeSetting: One byte that specifies the range setting.

Symbolic Name	Value
MDAC8_LOWRANGE	0
MDAC8_HIGHRANGE	1

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

MDAC8_SetPhase

Description:

Sets whether the internal φ1 and φ2 clocks are swapped or normal (default).

C Prototype:

```
void  MDAC8_SetPhase(BYTE bPhaseSetting)
```

Assembler:

```
mov   A, bPhaseSetting
lcall  MDAC8_SetPhase
```

Parameters:

bPhaseSetting: One byte that specifies normal or swapped phase.

Symbolic Name	Value
MDAC8_NORMALPHASE	0
MDAC8_SWAPPEDPHASE	1

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

MDAC8_WriteBlind

Description:

Immediately updates the output voltage to the indicated value.

C Prototypes:

```
// For OffsetBinary: (BYTE = unsigned char)
void  MDAC8_WriteBlind(BYTE bOutputValue)
// For TwosComplement: (CHAR = signed char)
void  MDAC8_WriteBlind(CHAR cOutputValue)
// For TwoByteSignAndMagnitude: (BYTE of bit flags)
void  MDAC8_WriteBlind2B(BYTE bLSB, BYTE bMSB)
```

Assembly:

```
; for OffsetBinary
mov   A, bOutputValue
lcall  MDAC8_WriteBlind
; for TwosComplement
mov   A, cOutputValue
lcall  MDAC8_WriteBlind
; for TwoByteSignAndMagnitude format:
mov   A, bLSB
mov   X, bMSB
lcall  MDAC8_WriteBlind2B
```

Parameters:

b/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	254
TwosComplement	-127	127
TwoByteSignAndMagnitude	3F18h	1F38h

Offset-binary values are positive numbers with the lowest output voltage represented by 0 and the highest by 254. 2's complement is the native signed format of the M8C processor. In TwoByteSignAndMagnitude format, high byte takes the form $00smmmmm_2$ and the low byte takes the form $00tmm000_2$, where 's' is the sign, 't' is the inverted sign, and 'm' represents magnitude bits; for positive numbers, s=0 and t=1.

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.

MDAC8_WriteStall

Description:

Possibly stalls the microprocessor until the beginning of Phi1, then updates the output voltage to the indicated value. Note that the API assumes that either interrupts are disabled or the maximum inter-

rupt latency is less than ACLKi (see the Forced Synchronization with Fast Update Clock timing diagram).

C Prototypes:

```
// For OffsetBinary: (BYTE = unsigned char)
void  MDAC8_WriteStall(BYTE bOutputValue)
// For TwosComplement: (CHAR = signed char)
void  MDAC8_WriteStall(CHAR cOutputValue)
// For TwoByteSignAndMagnitude: (BYTE of bit flags)
void  MDAC8_WriteStall2B(BYTE bLSB, BYTE bMSB)
```

Assembly:

```
; for OffsetBinary
mov   A, bOutputValue
lcall  MDAC8_WriteStall
; for TwosComplement
mov   A, cOutputValue
lcall  MDAC8_WriteStall
; for TwoByteSignAndMagnitude format:
mov   A, bLSB
mov   X, bMSB
lcall  MDAC8_WriteStall2B
```

Parameters:

b/cOutputValue: Identical in format and value range to the parameters described for the Write Blind entry point. bMSB and bLSB: Identical in format and value range to the parameters described for the Write Blind 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.

MDAC8_Stop

Description:

Powers the user module Off.

C Prototype:

```
void  MDAC8_Stop(void)
```

Assembly:

```
lcall  MDAC8_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 sample code creates a periodic, slowly descending sawtooth wave.

```
;;-------------------------------------------------------------------
;; Sample Code for the MDAC8
;; Generate a falling sawtooth wave
;;-------------------------------------------------------------------

export _main
include "m8c.inc"
include "MDAC8.inc"

      area bss (RAM)
bVal: blk  1                ; RAM for loop iteration variable
bMAXVAL: equ  255           ; Top of ramp plus 1
      area text (ROM, REL)

_main:                      ; (contains infinite loop; never returns)
      mov  A, MDAC8_LOWPOWER ; specify DAC's amplify power
      call MDAC8_Start      ; and turn it on.
Init:
      mov  [bVal], bMAXVAL  ; Start ramp from the top
RampDown:
      mov  A, [bVal]        ;
      dec  A
      call MDAC8_WriteStall
      dec  [bVal]           ; Bottom of ramp?
      jnz  RampDown         ; No, not yet.
      jmp  Init             ; Yes, re-initialize ramp and loop
                    ; forever


//-------------------------------------------------------------------
// C main line
//-------------------------------------------------------------------

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

    BYTE cVal;
    #define cMax 255


void main(void)
{
    // Insert your main routine code here.
    MDAC8_Start(MDAC8_LOWPOWER);
    while(1) //infinite loop
    {
        cVal = cMax;
        while(cVal > 0)
```

```
        {
            MDAC8_WriteStall(cVal--);
        }
    }
}
```

Configuration Registers

The API provides a complete interface to the MDAC8 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 MDAC8 switched capacitor LSB and MSB blocks.

Table 5. Block LSB: Register CR0

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

Sign uses a ‘1‘ for positive values (AGND up to RefHi) and a ‘0‘ for negative values (RefLow up to AGND). The default is ‘1‘. Note that this is opposite the sense used in the MSB Block. Use one of the Write functions in the API to change the sign value. Magnitude’s default is ‘0‘. Use one of the Write functions in the API to change this value.

Table 6. Block LSB: Register CR1

Switched Cap Type A								
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Value	0	1	0	0	0	0	0	0
Switched Cap Type B								
Bit	7	6	5	4	3	2	1	0
Value	1	0	0	0	0	0	0	0

Table 7. Block LSB: Register CR2

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

Analog Bus is disabled.

Table 8. Block LSB: Register CR3

Switched Cap Type A								
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Value	0	0	1	1	0	0	Power	
Switched Cap Type B								
Bit	7	6	5	4	3	2	1	0
Value	0	0	1	1	1	0	Power	

Power: The default is Off. Use the Start call in the API to set this value.

Table 9. Block MSB: Register CR0

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

Sign is set by the API Write routines. Its default is ‘0‘. Magnitude is changed by using one of the Write functions in the API. Its default is also ‘0‘.

Table 10. Block MSB: Register CR1

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

Table 11. Block MSB: Register CR2

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

Analog Bus is enabled or disabled at configuration time in the Device Editor.

Table 12. Block MSB: Register CR3

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

BMux is configured to select the connection from the LSB PSoC block. Power: 0=Off (default), 1=Low, 2=Medium, 3=Full. Use the Start call in the API to set this value.

Version History

Version	Originator	Description
2.2	DHA	Initial version.

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.