6-Bit Voltage Output Multiplying DAC Datasheet MDAC6 V 2.2

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/24/22xxx, CY8C23x33, CY8CLED04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28x43, CY8C28x52						
	0	0	1	112	0	1

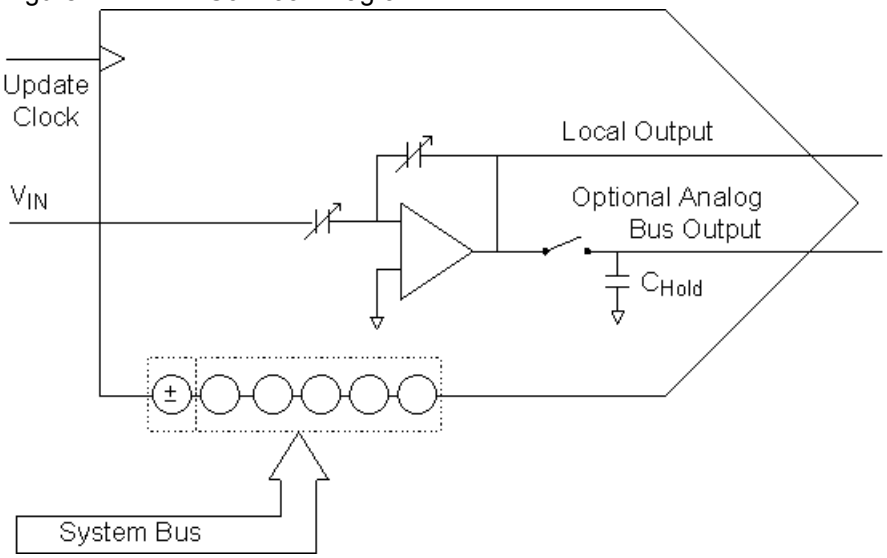
For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects.

Features and Overview

- 6-bit resolution
- Voltage output
- 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 250 ksps

The MDAC6 User Module is a 6-bit, four-quadrant multiplying DAC that scales input voltage with digital codes. The MDAC6 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. Offset compensation minimizes conversion error.

Figure 1. MDAC6 Block Diagram

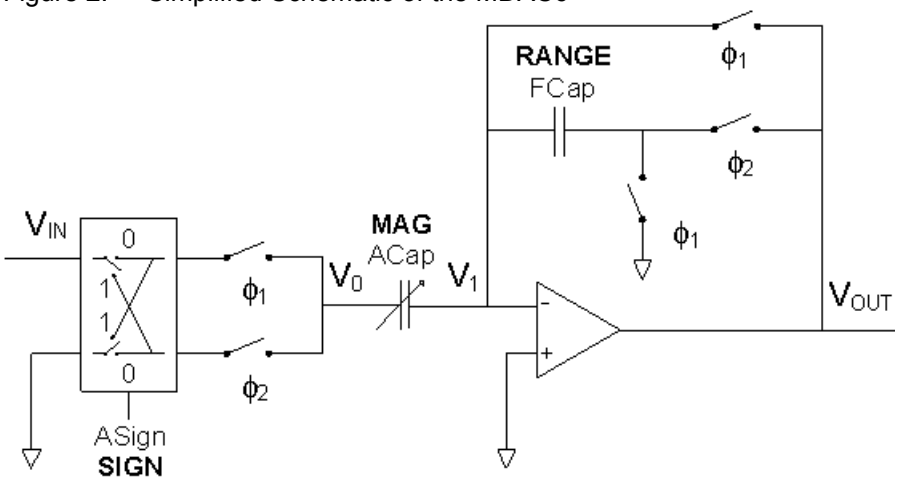


Functional Description

The MDAC6 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 -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. 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 MDAC6 User Module maps onto any single analog PSoC block. This block is designated MDAC. 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 zero to 31 units. The input voltage, which may be inverted by the ASign bit, is scaled at the output by the ratio of ACap to the feedback capacitor, FCap, which assumes values of 16 or 32 units (32 gains the input voltage by 1, 16 by 2).

Figure 2. Simplified Schematic of the MDAC6



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_{IN} to ACap while φ_1 is active, charging ACap to the input voltage minus AGND. After φ_1 goes inactive and as φ_2 becomes active, the input side of ACap is switched from V_{IN} to AGND, effectively inverting its sense with respect to AGND. As charge is shared between ACap and FCap, the opamp supplies an opposite charge sufficient to force the summing node voltage, V_1, to AGND.

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_{IN} 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. V_{out} charges both the load and the hold capacitor (C_{Hold} in the MDAC6 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.

Given the REFMUX parameter configured in the Device Editor to 2*BandGap ± BandGap, the following applies.

Equation 1

$$V_{OUT} = (V_{IN} - V)\frac{ACap}{FCap} + AGND$$

Equation 2

$$V_{OUT} = (V_{IN} - 2.6Volts)\frac{MAG}{RANGE} + 2.6Volt$$
$$(0 \le MAG \le 31)(RANGE \in 16.32)$$

Example

Given an input voltage of 1V provided by an external source and the GainRange parameter set to High then Equation 1 becomes as follows:

Equation 3

$$V_{Out} = 2.6Volts \pm 1.6Volts\left(\frac{MAG}{16}\right), 0 \le MAG \le 31$$

So if we are to assign a of 20 for the MAG variable and a negative sign, this results in an output of 0.6Volts. The solution is illustrated below:

Equation 4

$$V_{Out} = 2.6Volts - 1.6Volts\left(\frac{20}{16}\right) = 0.6V$$

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

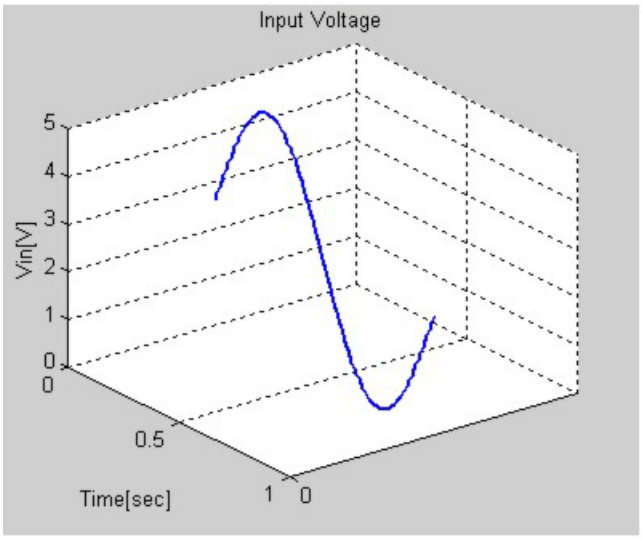


Figure 4. Output Voltage versus Input Code and Time, FCap=32

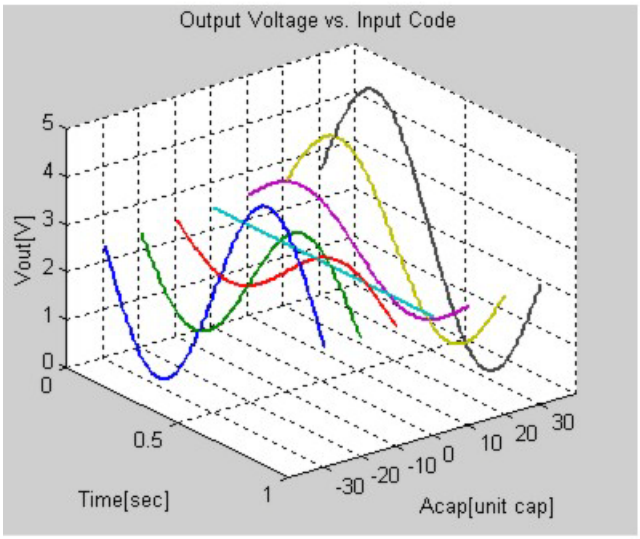
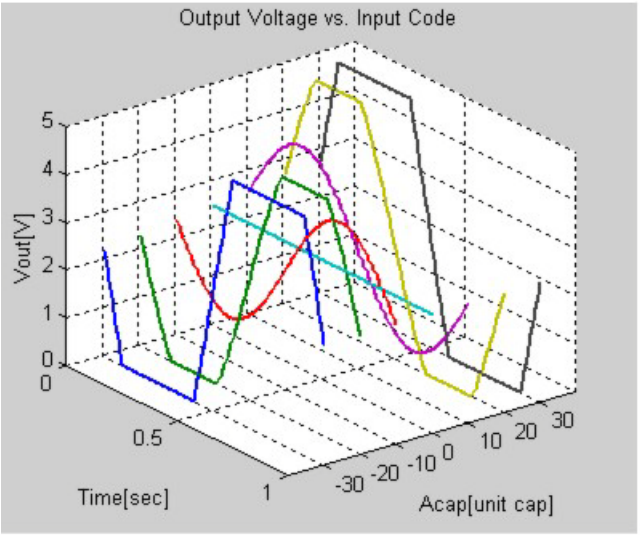


Figure 5. Output Voltage versus Input Code and Time, FCap=16



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

DC and AC Electrical Characteristics

CThe following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the table below, T_A = 25C 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.

Table 1. 5.0V MDAC6 DC and AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Resolution	--	6	Bits	
Linearity				
DNL	0.06	--	LSB	
INL	0.05	--	LSB	
Monotonic	YES	--		
Gain Error				
Including Reference Gain Error	3.4	--	%FSR	
Excluding Reference Gain Error[3]	0.45	--	%FSR	
V_{OS}, Offset Voltage	±3.3	--	mV	
Output Noise	4.6	--	mV rms	0 to 300 kHz
f_{clock}, Internal Update Rate[1]				

Parameter	Typical	Limit	Units	Conditions and Notes
Low Power	2 to 125	--	kHz	
Med Power	1 to 500	--	kHz	
High Power	1 to 800	--	kHz	
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 = 25C 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 MDAC6 DC and AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Resolution	--	6	Bits	
Linearity				
DNL	0.06		LSB	
INL	0.05		LSB	
Monotonic	YES			
Gain Error				
Including Reference Gain Error	2.9	--	%FSR	
Excluding Reference Gain Error[3]	0.3	--	%FSR	
V_{OS}, Offset Voltage	±3.6		mV	
Output Noise	2.1		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	150		µA	
Med Power	560		µA	
High Power	2150		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 data-Sheet).

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

Table 3. 5.0V MDAC6 DC and AC Electrical Characteristics

Parameter	Typical[1]	Limit[2]	Units	Conditions and Notes
Resolution	—	6	Bits	
Linearity				
DNL	.02	.05	LSB	
INL	.03	.08	LSB	
Monotonicity	—	½	Bit	
Gain Error	1.0	2.5	%FSR	
V_{OS}, Offset Voltage[3]	8	43	mV	
Output Noise				
Band Limited	.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[6]	40 kHz	—	kHz	
Operating Current[5]				
Low Power	125	—	µA	
Med Power	280	—	µA	
High Power	780	1000	µA	

Unless otherwise specified in the table below, all limits guaranteed for TA = 25C and Vdd = 3.3V. Also, fclock = 125 kHz, external AGND 1.50V, external VRef 0.80V, REFPWR = HIGH, SCPOWER = ON, PSoC block power HIGH.

Table 4. 3.3V MDAC6 DC and AC Electrical Characteristics

Parameter	Typical[1]	Limit[2]	Units	Conditions and Notes
Resolution	—	6	Bits	
Linearity				
DNL	.02	.04	LSB	
INL	.04	.09	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	.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[6]	40 kHz	—	kHz	
Operating Current[5]				
Low Power	100	—	µA	
Med Power	250	—	µA	
High Power	640	900	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. PSoC block current requirements exclusive of reference current.
6. Using ABUS output buffer.

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 MDAC6 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 that the magnitude value may be written without violating the setup time-requirement, see the Update Timing diagram below.

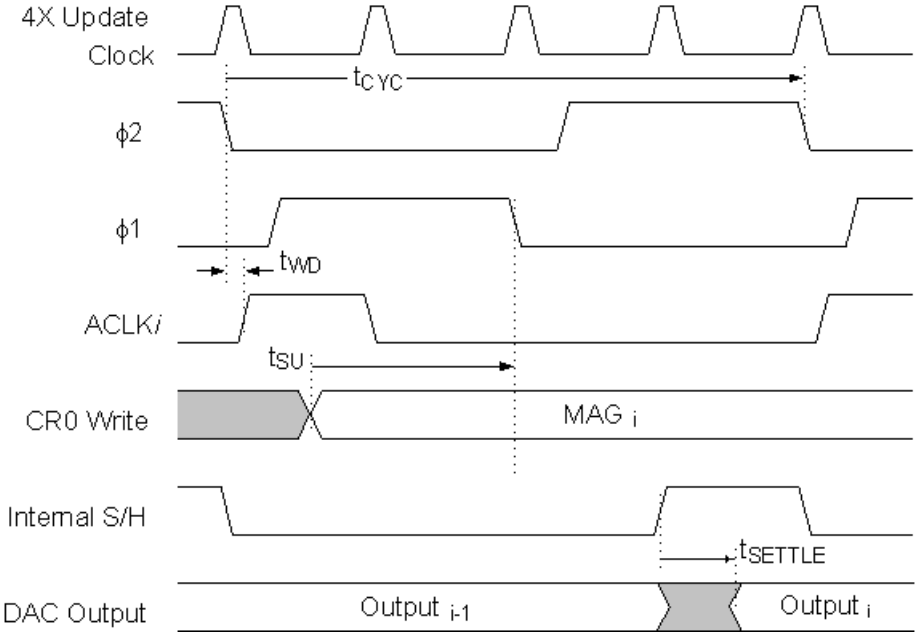
Failure to meet the setup time requirement will result in incorrect output data. If a write occurs while φ_2 is high, the output value may lie outside the voltage interval defined by the previous and desired outputs.

For a large class of applications, momentary (one update-cycle) deviations, such as those just described, are acceptable. In others, however, 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 proper access to the output-value registers. Hardware synchronization is invoked by writing the ASY_CR register. The next write to the corresponding control registers proceeds immediately, if ACLKi is active; if not, the CPU clock is stalled until ACLKi is

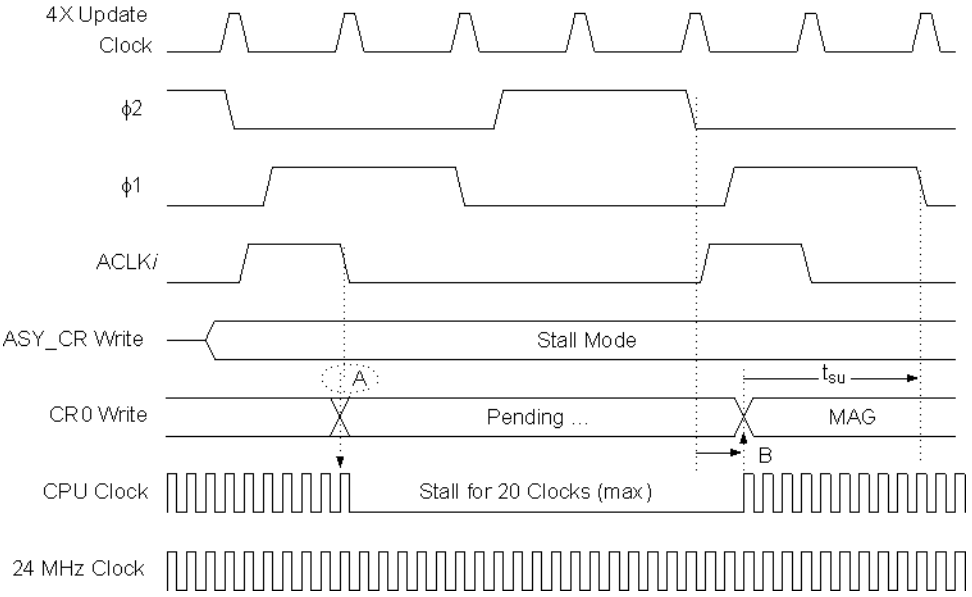
asserted. In either case, ACLKi is immediately de-asserted and does not again become active until the beginning of the next write cycle. To minimize the CPU cycles lost to the stall, the Update clock can be run at f_{Update}, even though actual update rates may be much slower. This process, which is illustrated in the Forced Synchronization with Fast Update Clock diagram, is most appropriate when interrupt timings and latencies are uncertain.

Figure 6. Update Timing



During the CPU stall, all analog and digital PSoC blocks function normally. The MOV instruction that writes the MDAC's CR0 register is simply suspended and, during this period, any interrupts become or remain pending.

Figure 7. Forced Synchronization with Fast Update Clock



Placement

The MDAC block maps freely onto any of the switched capacitor PSoC blocks in the device. However, if the MDAC 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 MDAC block must also be compatible with other user module blocks mapped into the same column of PSoC blocks.

Parameters and Resources

Update Clock

Configuring Update Clock resources involves three steps. First, provide 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, in this user module, covers the 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. If an external clock is used, it must first be connected through a digital PSoC block. All of the Timer, Counter, and Pulse-Width Modulator (PWM) User Modules make suitable choices for a rate generator, when V1 and V2 must be consigned to other uses.

Second, connect the clock source to the column clock generator by setting the CLKn mux in the Device Editor. Digital PSoC block outputs must also be connected through the ACLK0 or ACLK1 multiplexers.

Finally, select a value for the MDAC6 User Module parameter ClockPhase, by choosing Normal (the default value) or Swapped. This can synchronize the output of one analog PSoC block to the input of another. Switched capacitor analog PSoC blocks use φ_1 and φ_2 to acquire and transfer signal. Because the output of the MDAC 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 ClockPhase parameter to Swapped interchanges the roles of φ_1 and φ_2 within the block, so that the MDAC output will be valid when the other user module samples its input. Note that in Normal mode the input voltage is sampled during φ_1.

Data Format

The MDAC6 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.

Gain Range

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.

Vin Source

Selecting REFHI as the input voltage causes the MDAC6 to behave like the DAC6. Other input voltage selections require the configuration of an appropriate user module in the selected block.

Analog Output Bus

The MDAC block broadcasts its output to adjacent analog PSoC blocks. Choosing *AnalogOutBus_x* extends this set of possible connections to the outside world through the analog output buffer in column “x.” Selecting the bus, in this way, extends the local connections in several cases to blocks in the same column at the top and bottom of the array.

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.

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 MDAC6 User Module, write updated values, and disable the user module.

MDAC6_Start

Description:

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

C Prototype:

```
void  MDAC6_Start(BYTE bPowerSetting)
```

Assembler:

```
mov   A, bPowerSetting
lcall  MDAC6_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
MDAC6_OFF	0
MDAC6_LOWPOWER	1
MDAC6_MEDPOWER	2
MDAC6_FULLPOWER	3

Return Value:

None

Side Effects:

The DAC outputs will be driven. By default, the initial output 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.

MDAC6_Stop

Description:

Powers the user module Off.

C Prototype:

```
void  MDAC6_Stop(void)
```

Assembly:

```
lcall  MDAC6_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

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

MDAC6_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  MDAC6_SetPower(BYTE bPowerSetting)
```

Assembler:

```
mov   A, bPowerSetting
lcall  MDAC6_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 output voltage is AGND. Call one of the Write routines prior to calling “SetPower," if some other output value is required at power on. The A and X registers may be altered by this function.

MDAC6_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  MDAC6_SetOutputRange(BYTE bRangeSetting)
```

Assembler:

```
mov   A, bRangeSetting
lcall  MDAC6_SetOutputRange
```

Parameters:

RangeSetting: One byte that specifies the range setting.

Symbolic Name	Value
MDAC6_LOWRANGE	0
MDAC6_HIGHRANGE	1

Return Value:

None

Side Effects:

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

MDAC6_SetPhase

Description:

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

C Prototype:

```
void  MDAC6_SetPhase(BYTE bPhaseSetting)
```

Assembler:

```
mov   A, bPhaseSetting
lcall  MDAC6_SetPhase
```

Parameters:

bPhaseSetting: One byte that specifies normal or swapped phase.

Symbolic Name	Value
MDAC6_NORMALPHASE	0
MDAC6_SWAPPEDPHASE	1

Return Value:

None

Side Effects:

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

MDAC6_WriteBlind

Description:

Immediately updates the output voltage to the indicated value.

C Prototypes:

```
void  MDAC6_WriteBlind(char cOutputValue)
```

Assembler:

```
mov   A, cOutputValue
lcall  MDAC6_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 M8C. Offset-binary values are positive numbers. The lowest output voltage is represented by zero and the highest by 62 if the input voltage is greater than AGND. The lowest output voltage is represented by 62 and the highest by Zero if the input voltage is less than AGND. 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.

MDAC6_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  MDAC6_WriteStall (char cOutputValue)
```

Assembler:

```
mov   A, cOutputValue
lcall  MDAC6_WriteStall
```

Parameters:

cOutputValue: Identical in format and value range to the OutputValue 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.

Sample Firmware Source Code

The sample code creates a periodic, slowly descending sawtooth wave.

```
;;-------------------------------------------------------------------
;; Sample Code for the MDAC6
;; Generate a falling sawtooth wave
;;-------------------------------------------------------------------

export _main
include "m8c.inc"
include "MDAC6.inc"

area bss (RAM)
cVal:   blk  1                  ; RAM for loop iteration variable
cMAX:   equ  63                 ; Top of ramp plus 1
area text (ROM, REL)

_main:                          ; (contains infinite loop; never returns)
    mov   A, MDAC6_LOWPOWER     ; specify MDAC's amplify power
    call  MDAC6_Start           ; and turn it on.
Init:
    mov   [cVal], cMAX          ; Start ramp from the top
```

```
RampDown:
    mov   A, [cVal]
    dec   A                     ; Note, data is offset binary in [0..62]
    call  MDAC6_WriteStall
    dec   [cVal]                ; 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 63


void main(void)
{
    // Insert your main routine code here.
    MDAC6_1_Start(MDAC6_1_LOWPOWER);
    while(1) //infinite loop
    {
        cVal = cMax;
        while(cVal > 0)
        {
            MDAC6_1_WriteStall(cVal--);
        }
    }
}
```

Configuration Registers

The API provides a complete interface to the MDAC6 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 MDAC6 switched capacitor DAC block.

Table 5. Block DAC ASAxxCR0 or ASBxxCR0: Register CR0

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

Range is set to 32 units for low gain range and 16 units for high gain range. It is modified by way of the “SetOutputRange" API calls. 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 6. Block DAC ASAxxCR1 or ASBxxCR1: Register CR1

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

Input: Choose REFHI to become like DAC6 or choose the output of another block (such as AnalogMux). It is set while in the user module Placement mode of the Device Editor.

Table 7. 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

Analog Bus 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 8. 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 9. 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
2.2	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.