6-Bit 电压输出乘法 DAC 数据手册 MDAC6 V 2.2

Copyright © 2012 Cypress Semiconductor Corporation. All Rights Reserved.

资源	PSoC® 模块			API 内存 （字节）		引脚 （每个外部 I/O）
	数字	模拟 CT	模拟 SC	Flash （闪存）	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

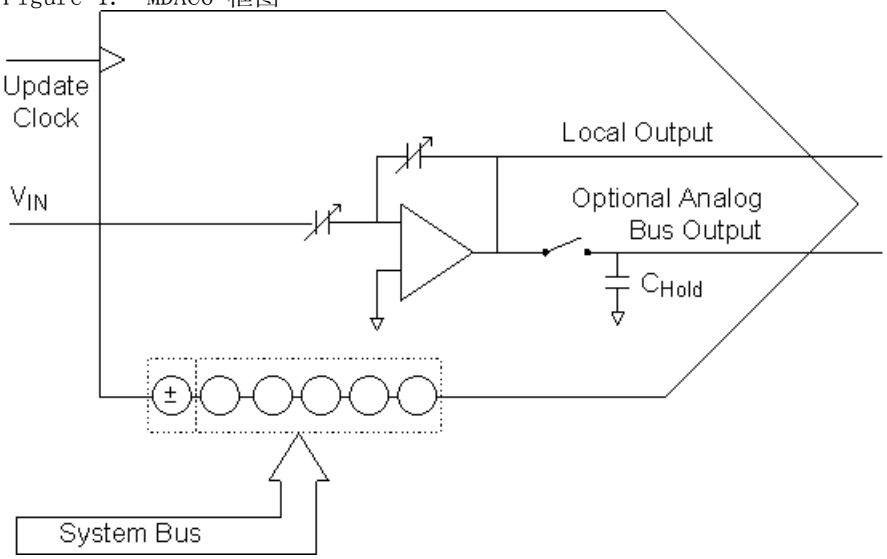
如需一个或多个使用此用户模块且完全配置的功能性示范项目，请转到 www.cypress.com/psocexampleprojects.

功能和概述

- 6-bit 8 位分辨率
- 电压输出
- 四象限乘法
- 2 的补码、偏移二进制和正负 / 大小输入数据格式
- 模拟总线和外部输出的采样和保持
- 更新速率高达 250 ksps

MDAC6 用户模块是使用数字码标度输入电压的 6-bit、四象限乘法 DAC。 MDAC6 以高达 250K 每秒的更新速率将数字输入转换为输出电压。 应用程序编程接口 （API） 支持偏移二进制、原码以及补码数据格式。 偏移补偿最大程度地减小了转换误差。

Figure 1. MDAC6 框图

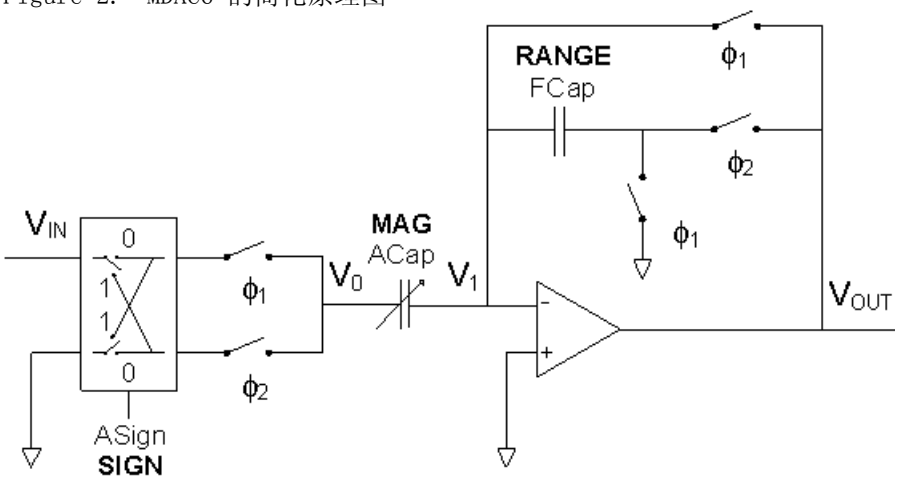


功能描述

MDAC6 用户模块使用数字代码乘以模拟输入电压。 数字代码显示为从 -31 到 +31 的二进制补码或原码格式的数字。 另外，输入代码也可以使用从 0 到 62 的偏移二进制数字。 这意味着输出电压的步长为全量程输出范围的 1/63，而不是较为常见的 1/64。 在原码格式下，输入代码 “-0” 会通过用户模块 API 转换为 “+0”。 输入和输出的参考电压是 AGND，AGND 的数值由系统参数 RefMux 的设置来决定。 输入电压可乘以 1 或者 2，具体取决于为 SetOutputRange API 函数选定的值。

MDAC6 用户模块可放置到任意一个 PSoC 模拟模块。 该模块被命名为 MDAC。 模块内部的操作基于原码格式完成。 五个数量级位设置了 ACap 的值，以下简化原理图中显示了二进制加权电容的阵列。 ACap 采用从 0 到 31 个单位范围内的值。 可能被 ASign 位反转的输入电压，在输出端按 ACap 比例传给反馈电容 FCap，而 FCap 采用 16 或 32 单位的值 （32 时输入电压增益 1 倍，16 时 增益 2 倍）。

Figure 2. MDAC6 的简化原理图



SIGN 位通过交叉开关与时钟信号控制的开关对极性进行控制。 Φ_1 和 Φ_2。 这两个信号周期相同，相位相反。 Φ_1 和 Φ_2 为 “ 部分重叠 ”。 也就是说， 每两个脉冲之间存在一个较短时间，在这段时间内二者均处于非活动状态。 这保证了 Φ_1 和 $\Phi 2$ 以先开后合的方式打开与关闭各自的开关。 时钟发生器在每个模拟列中将源时钟与 24 MHz 振荡器同步，并四分频生成相应的信号。

当输入码为正 （SIGN 位是 0）时，交叉开关在为活动状态时，将 V_{IN} 连接到 ACap， Φ_1 将 ACap 充电至输入电压减去 AGND。 在 Φ_1 变为非活动状态 Φ 而 $_2$ 为活动状态后，ACap 的输入侧从 V_{IN} 变为 AGND。 由于 ACap 和 FCap 共享电荷，运算放大器提供了足够的相反电荷，迫使求和节点电压 V_1 变为 AGND。

因此，对于正向输入代码，将发生两次电压反转。 当 ACap 的源终端切换至 AGND 时发生首次反转。 由于 ACap 连接到反相输入中，运算放大器本身进行二次反转。 针对负向输入代码的分析是类似的。 主要的区别在于 V_{IN} 直接施加 Φ 在 $_2$ 间， Φ 而非 $_1$。 这样 ACap 电荷不会发生有效反转，仅有的反转由运算放大器提供。

硬件会在每个更新周期执行偏移补偿。 由 Φ_1 和 Φ_2 控制的开关配置运算放大器， Φ 在 $_1$ 过程中为单位增益跟随器。 在此配置中，偏移电压出现在求和节点上，为 ACap 和 FCap 充电。 由于电路在 Φ_2 中重新进行了配置，因此可以反转这些电容中的电荷偏移量，从而有效抵消偏移电压。

在各更新周期，V_{out} 都在运算放大器偏移电压 Φ（在 $_1$ 过程中）及希望的电压 Φ（在 $_2$ 过程中稳定）之间转换，这是偏移补偿的直接后果。 降低提高精度带来的代价的一种方法是使用与输出总线相关联的采样与保持电路。 在 $_2$ 的后半部分，V_{out} 将对加载和保持电容器 （MDAC6 框图中的 C_{Hold}）进行充电 Φ。 在该周期结束时，C_{Hold} 将会与运算放大器输出分离。 每个模拟输出总线由带有适当较高输入阻抗的模拟输出缓冲区提供支持。

如果 REFMux 参数由器件编辑器配置为 2*BandGap ± BandGap，则以下公式成立。

Equation 1

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

Equation 2

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

示例

假设外部源提供的输入电压为 1V，并且 增益范围 (GainRange) 设为 High，则等式 1 变成如下所示：

Equation 3

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

因此，如果我们要分配一个值 20 给 MAG 变量及一个负号，会导致输出为 0 6Volts。 以下是解决办法：

Equation 4

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

计算得到的值是理想值，该值很可能会根据系统噪声和芯片偏移而有所不同。

四象限乘法的意思是输入电压和输入代码均可导致输出电压为正或者为负。 请参见以下 3 幅图。

Figure 3. 输入电压与时间的关系

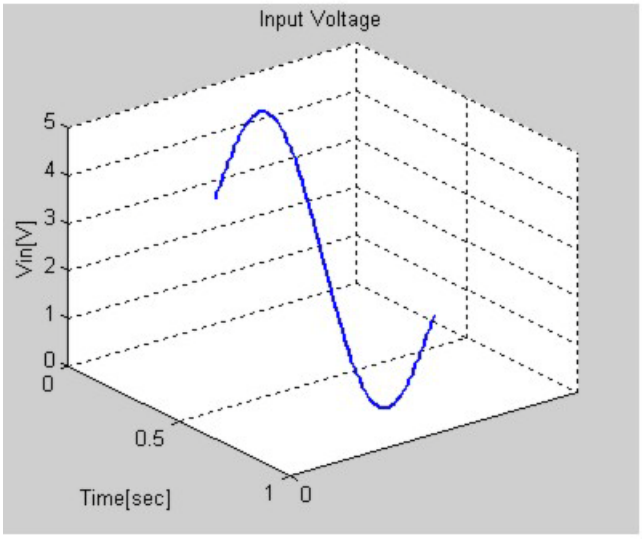


Figure 4. 输出电压与输入代码和时间，FCap=32

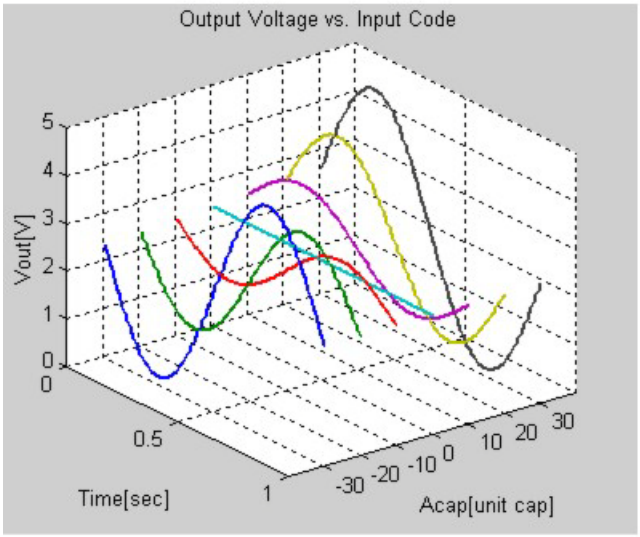
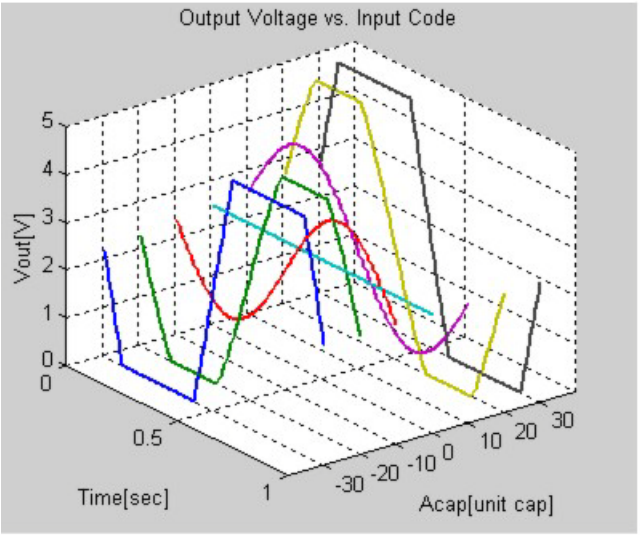


Figure 5. 输出电压与输入代码和时间，FCap=16



输入电压必须减小，以防止输出钳位，如上图所示。

直流和交流电气特性

CThe following values are indicative of expected performance and based on initial characterization data. 除非下表中另有说明，否则 T_A = 25°C，Vdd = 5V。并且，f_{clock} = 125 kHz，外部 AGND 2.50V，外部 V_{Ref} 1.23V，REFPWR = HIGH，SCPOWER = ON，PSoC 模块功耗为 “HIGH”。

Table 1. 5.0V MDAC6 直流和交流电气特性

参数	典型值	限制	单位	条件和注释
分辨率	--	6	位	
线性度				
差分非线性 (DNL)	0.06	--	LSB	
积分非线性 (INL)	0.05	--	LSB	
单调性	是	--		
增益误差				
包括参考增益误差	3.4	--	%FSR	
不包括参考增益误差[3]	0.45	--	%FSR	
V_{OS}，偏移电压	±3.3	--	mV	
输出噪声	4.6	--	mV rms	0 到 300 kHz
f_{clock}，内部更新速率[1]				

参数	典型值	限制	单位	条件和注释
低功耗	2 到 125	--	kHz	
中功耗	1 到 500	--	kHz	
高功耗	1 到 800	--	kHz	
工作电流 [2]				
低功耗	155	--	µA	
中功耗	585	--	µA	
高功耗	2225	--	µA	

The following values are indicative of expected performance and based on initial characterization data. 除非下表中另有说明，否则 TA = 25C，Vdd = 3.3V。并且，fclock = 125 kHz，外部 AGND 1.50V，外部 VRef 0.8V，REFPWR = HIGH，SCPOWER = ON，PSoC 模块功耗为 “HIGH”。

Table 2. 3.3V MDAC6 直流和交流电气特性

参数	典型值	限制	单位	条件和注释
分辨率	--	6	位	
线性度				
差分非线性 (DNL)	0.06		LSB	
积分非线性 (INL)	0.05		LSB	
单调性	是			
增益误差				
包括参考增益误差	2.9	--	%FSR	
不包括参考增益误差 [3]	0.3	--	%FSR	
V_{OS}，偏移电压	±3.6		mV	
输出噪声	2.1		mV rms	0 到 300 kHz
f_{clock}，内部更新速率 [1]				
低功耗	2 到 125	--	kHz	
中功耗	1 到 500	--	kHz	
高功耗	1 到 800	--	kHz	
工作电流 [2]				
低功耗	150		µA	
中功耗	560		µA	
高功耗	2150		µA	

电气特性说明

1. 对 Φ_1、 Φ_2 的限制；针对宽频带噪声增加 3dB 而指定。
2. 不包括所有模拟模块均通用的参考模块功耗 （请参阅 PSoC 系列基本介绍）。

除非下表中另有说明，否则在以下条件下保证所有限值：TA = 25° C，Vdd = 5V。并且，fclock = 125 kHz，外部 AGND 2.50V，外部 VRef 1.23V，REFPWR = HIGH，SCPOWER = ON，PSoC 模块功耗为 “HIGH”。

Table 3. 5.0V MDAC6 直流和交流电气特性

参数	典型值[1]	限值[2]	单位	条件和注释
分辨率	—	6	位	
线性度				
差分非线性 (DNL)	.02	.05	LSB	
积分非线性 (INL)	.03	.08	LSB	
单调性	—	½	位	
增益误差	1.0	2.5	%FSR	
V_{OS}，偏移电压[3]	8	43	mV	
输出噪声				
频带限值	.3	1	mV rms	0 到 10 kHz
宽频带	7	10	mV rms	0 到 300 kHz
f_{clock}，内部更新速率[4]	—	32 到 333	kHz	
V_{in} 频带宽[6]	40 kHz	—	kHz	
工作电流[5]				
低功耗	125	—	µA	
中功耗	280	—	µA	
高功耗	780	1000	µA	

除非下表中另有说明，否则在以下条件下保证所有限值：TA = 25° C，Vdd = 3.3V。并且，fclock = 125 kHz，外部 AGND 1.50V，外部 VRef 0.80V，REFPWR = HIGH，SCPOWER = ON，PSoC 模块功耗为 “HIGH”。

Table 4. 3.3V MDAC6 直流和交流电气特性

参数	典型值[1]	限值[2]	单位	条件和注释
分辨率	—	6	位	
线性度				
差分非线性 (DNL)	.02	.04	LSB	
积分非线性 (INL)	.04	.09	LSB	
单调性	—	½	位	
增益误差	1.0	2.5	%FSR	

参数	典型值[1]	限值[2]	单位	条件和注释
V_{OS}，偏移电压[3]	7	31	mV	
输出噪声				
频带限值	.3	1	mV rms	0 到 10 kHz
宽频带	7	10	mV rms	0 到 300 kHz
f_{clock}，内部更新速率[4]	—	32 到 333	kHz	
V_{in} 频带宽[6]	40 kHz	—	kHz	
工作电流[5]				
低功耗	100	—	µA	
中功耗	250	—	µA	
高功耗	640	900	µA	

电气特性说明

1. 典型值表示统计平均值加 1σ。
2. 限值通过测试或统计分析得到。
3. 对外部 AGND 的二进制补码零标度偏移，不包括模拟输出缓冲区偏移误差。
4. 对 Φ_1、Φ_2 的限制；针对宽频带噪声增加 3dB 而指定。
5. PSoC 模块电流要求唯一的参考电流。
6. 使用 ABUS 输出缓冲器。

时序

为了生成针对部分重叠相位时钟、Φ_1 和 Φ_2 的时序信号和合适的占空比，模拟列时钟电路采用了一对二分压电路。 因此，为 MDAC6 选定的时钟源的运行速率至少是所需最大更新速率的 4 倍。 除在内部用于去除输出短时脉冲的采样与保持信号外，还会生成 “ 就绪 ” 信号。 Ready 表示在不违反建立时间的要求下写入的幅度值，见下面的更新时序图。

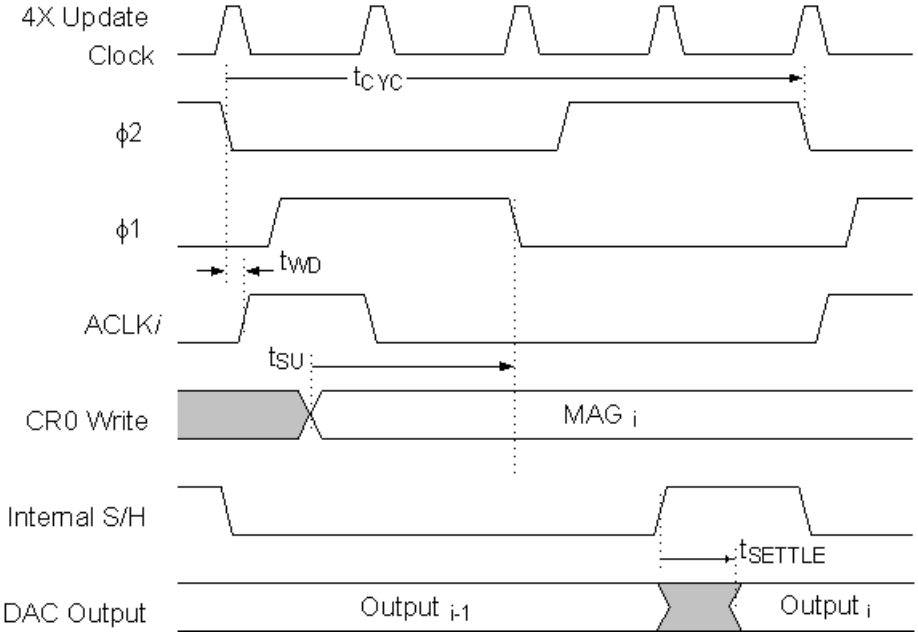
达不到建立时间要求会导致错误的输出数据。 如果在 Φ_2 为高时写入，则输出值可能会在之前输出和所需输出确定的电压区间之外。

对于很多应用，上述瞬时 （一个更新周期内的）偏差是可以接受的。 对于其他应用程序，要求必须更为严格。 例如低失真波形发生器。 硬件同步是一种确定寄存器更新时序的方法，此方法可用于避免这样的问题，并且受 API 中以单词 “Stall” 结尾的进入点直接支持。

硬件同步可保证对输出值寄存器进行尽早访问。 写入 ASY_CR 寄存器将会触发硬件同步。 如果 ACLKi 处于活动状态，下次则会立即写入相应控制寄存器；如果不处于活动状态，则 CPU 时钟将通过 IO 写入中途停顿，直到 ACLKi 被置为有效。 这两种情况下，ACLKi 被立即置为无效，且不会再次变为活动状态，直至

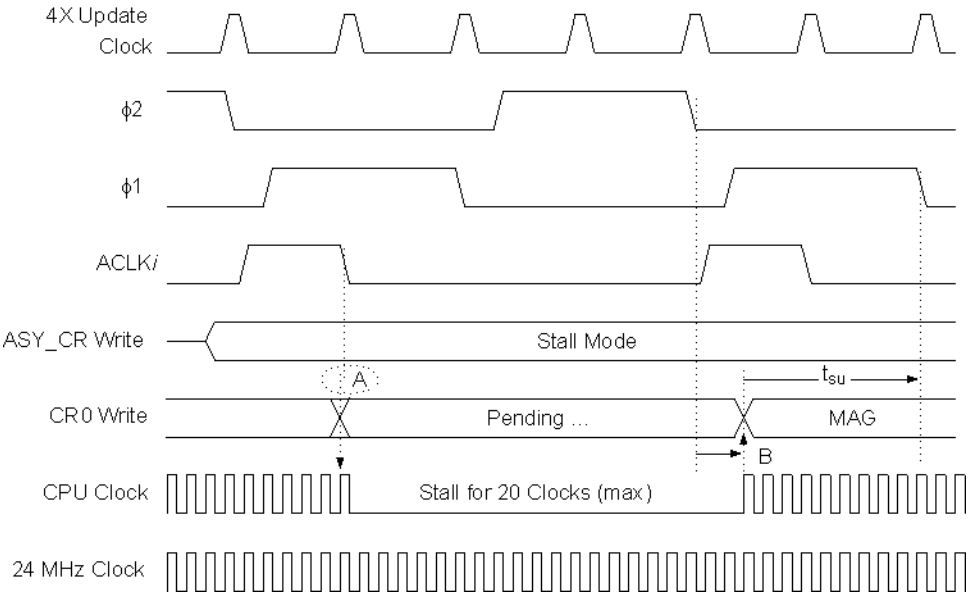
下一写入周期开始为止。 为降低因停顿丢失的 CPU 周期，Update 时钟可运行在 f_{Update}，即使实际更新速率可能低得多。 这个显示在快速更新时钟强制同步图中的过程，在中断时序和延迟不确定时是最适合的。

Figure 6. 更新时序



CPU 停止期间，所有模拟与数字 PSoC 模块功能正常运行。 延迟期间，用于指示写入 MDAC 的 CR0 寄存器的 MOV 指令仅暂停，所有中断会进入或保持待处理状态。

Figure 7. 与快速更新时钟的强制同步



放置

MDAC 模块可自由映射到该器件中的任何开关电容 PSoC 模块中。 不过，如果 MDAC 输出针对模拟输出总线启用，则必须多加注意，以确保没有其他用户模块试图驱动相同的总线。 在选择放置位置时需要额外注意的是，MDAC 模块使用的时钟还必须与映射到 PSoC 模块同一列中的其他用户模块相兼容。

参数和资源

更新时钟

配置更新时钟资源包括三个步骤。 第一，提供模拟列时钟发生器的时钟源。 列时钟发生器将输入时钟四分频，产生 Φ_1 和 Φ_2，因此源时钟必须以希望模拟输出更新速率四倍的速率运行。 在用户模块的 “ 时序 ” 章节讨论了与选择更新时钟频率相关的问题。 时钟源的选择包括 V1 和 V2 分频器以及任一 PSoC 数字模块的输出。 如果使用外部时钟，必须先通过 PSoC 数字模块连接。 当 V1 和 V2 必须指派用于其他用途，所有定时器、计数器和脉冲宽度调制器 (PWM) 用户模块均是适合的选择。

第二，通过在器件编辑器中设置 CLKn mux 将时钟源连接到列时钟发生器。 PSoC 数字模块输出还必须通过 ACLK0 或 ACLK1 复用器进行连接。

最后，通过选择 “ 标准 ” （默认值）或 “ 交换 ” 选项选择 MDAC6 用户模块参数 ClockPhase 的值。 这样可将其他模块输入与一个 PSoC 模块的输出同步。 开关电容 PSoC 模块使用 Φ_1 和 Φ_2 获取和传输信号。 由于 MDAC 模块的输出仅在 Φ_2 过程中有效，在把它连接到在 Φ_1 过程中对其输入采样的另一个用户模块时，会发生问题。 设置 ClockPhase 参数为 Swapped，可以使模块内的 Φ_1 和 Φ_2 作用互换，因此，当另一用户模块对其输入进行采样时，MDAC 输出将有效。 （请注意，在 “ 正常 ” 模式下，输入电压在 Φ_1 中进行取样。）

数据格式

MDAC6 用户模块 API 处理三种不同的数据格式：偏移二进制、补码和原码。 本用户模块 API 部分中的 WriteBlind 入口点描述了这些规范和各种格式相关的值范围。

增益范围 (Gain Range)

对于指定输入代码、输入电压和 AGND，从低到高更改增益范围将会相应增大输出电压。 请注意，高增益范围内的输入电压范围实际上是低增益范围内输入电压范围的一半。

输入电压源

选择 REFHI 作为输入电压源将会使 MDAC6 的功能与 DAC6 相似。 其他输入电压选项需要配置所选模块中相应参数选项。

模拟输出总线

MDAC 模块可以将其输出传递到相邻的 PSoC 模拟模块。 选择 *AnalogOutBus_x* ，可通过 “x” 列中的模拟输出缓冲器将这组可能的连接延伸到外部。 以这种方式选择总线，在某些情况下可将本地连接延伸至阵列顶部和底部同一列的模块中。

每个 PSoC 开关电容模块均采用在 Φ_2 中对总线启用信号进行采样的电路。 这样可消除在自动归零操作过程中出现的电压摆幅。

Note 将 AnalogBus 设置为 “ 启用 ” 并将 ClockPhase 设置为 “ 交换 ” 将会禁用采样和保持功能。 在这种情况下，总线输出将会在 Φ 1 中的 AGND （加上偏移电压）和 Φ 2 中的所需输出之间进行交替，从而映射本地 PSoC 模块输出。

应用程序编程接口

提供的应用程序编程接口 (API) 子程序作为用户模块的一部分，允许设计人员能够采用更高级的方式处理模块。 本节指定每个函数的接口，以及 “ 引用 ” 文件所提供的相关常量。

Note

在这里，如同所有用户模块 API 中的一样，A 和 X 寄存器的值可能在调用 API 函数时发生更改。 如果在调用后需要 A 和 X 的值，则调用函数负责在调用前保留 A 和 X 的值。 选择此 “ 寄存器易失 ” 策略是为了提高效率，自 PSoC Designer 1.0 版起已强制使用此策略。 C 编译器自动遵循此要求。 汇编语言程

序员也必须确保其代码遵守这一策略。 虽然一些用户模块的 API 函数可以保留 A 和 X 不变，但是无法保证它们将来也会如此。

对于大型储存器模型器件， 保存 CUR_PP、IDX_PP、MVR_PP 以及 MVW_PP 寄存器中的所有值也是程序调用者的职责。 尽管部分寄存器现在可能不可修改，但是无法保证在将来的版本中也会如此。

入口点用于初始化 MDAC6 用户模块、写入更新值和禁用用户模块。

MDAC6_Start

说明：

对此用户模块执行所有必需的初始化，并设置开关电容 PSoC 模块的功耗水平。

C 原型：

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

汇编程序：

```
mov   A, bPowerSetting
lcall  MDAC6_Start
```

参数：

bPowerSetting： 1 个用于指定功耗水平的字节。 在复位和配置之后，会关闭分配给 DAC 模块的 PSoC 模块的电源。 下表给出了 C 语言和汇编语言中提供的符号名称及其相关值。

符号名	值
MDAC6_OFF	0
MDAC6_LOWPOWER	1
MDAC6_MEDPOWER	2
MDAC6_FULLPOWER	3

返回值：

None

副作用：

DAC 输出将被驱动。 默认情况下，初始输出电压为 AGND。 如果通电时要求其他输出值，在调用 “ 启动 ” 前先调用一个写入例程。 此函数可能会更改 A 和 X 寄存器。

MDAC6_Stop

说明：

断开用户模块的电源。

C 原型：

```
void  MDAC6_Stop(void)
```

汇编：

```
lcall  MDAC6_Stop
```

参数：

None

返回值：

None

副作用：

将不会驱动输出。 此函数可能会更改 A 和 X 寄存器。

MDAC6_SetPower

说明：

设置 DAC 开关电容 PSoC 模块的功耗水平。 可用于打开和关闭模块。

C 原型：

```
void  MDAC6_SetPower(BYTE bPowerSetting)
```

汇编程序：

```
mov   A, bPowerSetting
lcall  MDAC6_SetPower
```

参数：

bPowerSetting: 与 “ 启动 ” 入口点使用的 “ 电源设置 ”(PowerSetting) 参数相同。

返回值：

None

副作用：

将会运行 MDAC 输出。 默认情况下，初始输出电压为 AGND。 如果通电时要求其他输出值，在调用 “SetPower” 前先调用一个写入例程。 此函数可能会更改 A 和 X 寄存器。

MDAC6_SetOutputRange

说明：

通过将 FCap 设置为 32 （低范围：gain=1）或 16 （高范围：gain=2），为 MDAC 开关电容 PSoC 模块设置其中一个范围。

C 原型：

```
void  MDAC6_SetOutputRange(BYTE bRangeSetting)
```

汇编程序：

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

参数：

RangeSetting: 用于指定范围设置的一个字节。

符号名	值
MDAC6_LOWRANGE	0
MDAC6_HIGHRANGE	1

返回值：

None

副作用：

此函数可能会更改 A 和 X 寄存器。

MDAC6_SetPhase

说明：

设置 Φ1 和 Φ2 个 内部时钟工作在 “ 交换 ” 模式还是 ″ 正常 ″（默认） 模式。

C 原型：

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

汇编程序：

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

参数：

bPhaseSetting： 用于指定正常或交换相位的一个字节。

符号名	值
MDAC6_NORMALPHASE	0
MDAC6_SWAPPEDPHASE	1

返回值：

None

副作用：

此函数可能会更改 A 和 X 寄存器。

MDAC6_WriteBlind

说明：

立即将输出电压更新为指定值。

C 原型：

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

汇编程序：

```
mov   A, cOutputValue
lcall  MDAC6_WriteBlind
```

参数：

cOutputValue： 用于指定输出电压的一个字节。 允许的数值范围对应于 DataFormat （数据格式）的选择数值，如下表所示。

数据格式	最小值	最大值
偏移二进制 (OffsetBinary)	0	62
二进制补码 (TwosComplement)	-31	31
原码 (SignAndMagnitude)	-31	31

“ 二进制补码 ”(TwosComplement) 与 “ 偏移二进制 ”(OffsetBinary) 使用 M8C 的本地二进制补码格式。 偏移二进制值是正数。 如果输入电压大于 AGND，最低输出电压用 0 表示，最高输出电压用 62 表示。 如果输入电压小于 AGND，最低输出电压用 62 表示，最高输出电压用 0 表示。 在 “ 原码 ”(SignAndMagnitude) 格式中，要求字节为二进制格式 “00smmmmm”，其中 “mmmmm” 为大小，“s” 为符号。 编码 s 时，使用 0 代表正数，1 代表负数。

返回值：

None

副作用：

输出时可能出现短时脉冲的原因将在本用户模块的 “ 时序 ” 章节中讨论。 此函数可能会更改 A 和 X 寄存器。

MDAC6_WriteStall

说明：

可以使微处理器停顿，直至 Φ_1 开始，然后将输出电压更新为指定值。 请注意，API 假设已禁用中断，或者最大中断延迟小于 ACLKi。

C 原型：

```
void  MDAC6_WriteStall (char cOutputValue)
```

汇编程序：

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

参数：

cOutputValue： 与 WriteBlind 入口点所述 OutputValue 参数的格式和值范围相同。

返回值：

None

副作用：

如果 ACLKi 处于非活动状态（“i” 是模拟 PSoC 模块映射到的列），微处理器的 CPU 时钟已禁用，直到 Φ_2 变为非活动状态，约需四分之三个更新周期 （加上两个 CPU 时钟）。 请注意，停止间隔期间不会识别中断。 此函数可能会更改 A 和 X 寄存器。

固件源代码示例

此示例代码创建了周期性缓慢下降的锯齿波。

```
;;-------------------------------------------------------------------
;; 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--);
        }
    }
}
```

配置寄存器

API 为 MDAC6 用户模块提供完整接口。 直接写入到配置寄存器是更新输出的另一种方式。 不论采用哪种方法，都存在时序注意事项，必须了解这些情况以防止出现输出短时脉冲。 下面的寄存器用于 MDAC6 开关电容 DAC 模块。

Table 5. 模块 DAC ASAxxCR0 或 ASBxxCR0： 寄存器 CR0

位	7	6	5	4	3	2	1	0
值	范围	0	码值					

对于低增益范围，范围设置为 32 个单位，而对于高增益范围，则设置为 16 个单位 . 可通过在 API 中调用 “ 设置 (Set)OutputRange” 进行修改。 在执行复位与重新配置后 “ 原码 ” 会设置为中等量程 (AGND)。 可通过在 API 中调用 “ 写入 ” 进行修改。

Table 6. 模块 DAC ASAxxCR1 或 ASBxxCR1： 寄存器 CR1

位	7	6	5	4	3	2	1	0
值	输入			0	0	0	0	0

输入： 选择 REFHI 使其类似于 DAC6，或选择另一模块的输出 （例如 AnalogMux）。 在处于器件编辑器的用戶模塊放置模式時，設置它。

Table 7. 模块 DAC ASAxxCR2 或 ASBxxCR2： 寄存器 CR2

位	7	6	5	4	3	2	1	0
值	模拟总线	0	1	0	0	0	0	0

“ 模拟总线 ”(AnalogBus) 可决定 DAC PSoC 模是否会驱动总线。 此位域的值取决于在器件编辑器的用户模放置模式下对同名参数所作的选择。

Table 8. 模块 DAC ASAxxCR3 或 ASBxxCR3： 寄存器 CR3

位	7	6	5	4	3	2	1	0
值	0	0	1	1	0	0	功耗	

在器件复位与配置后 “ 电源 ” 会设置为 “ 关闭 ”。 可通过在 API 中调用 “ 启动 ”、“ 设置电源 ”(SetPower) 或 “ 停止 ” 入口点来进行修改。

Table 9. 全局寄存器 ASY_CR

位	7	6	5	4	3	2	1	0
值	0	0	0	0	0	0	0	1

API 在需要时写入到此寄存器，并停止 CPU 以保证符合输出更新时序的要求。

版本历史记录

版本	创作者	说明
2.2	DHA	添加了版本历史

Note PSoC Designer 5.1 在所有用户模基本介绍中都引入了 “ 版本历史 ”。 本数据表详细介绍了当前和先前用户模版本之间的区别。

Copyright © 2012 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.