6-Bit 电压输出 DAC 数据手册 DAC6 V 4.3

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	闪存	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
CY8C26/25xxx	0	0	1	61	0	1

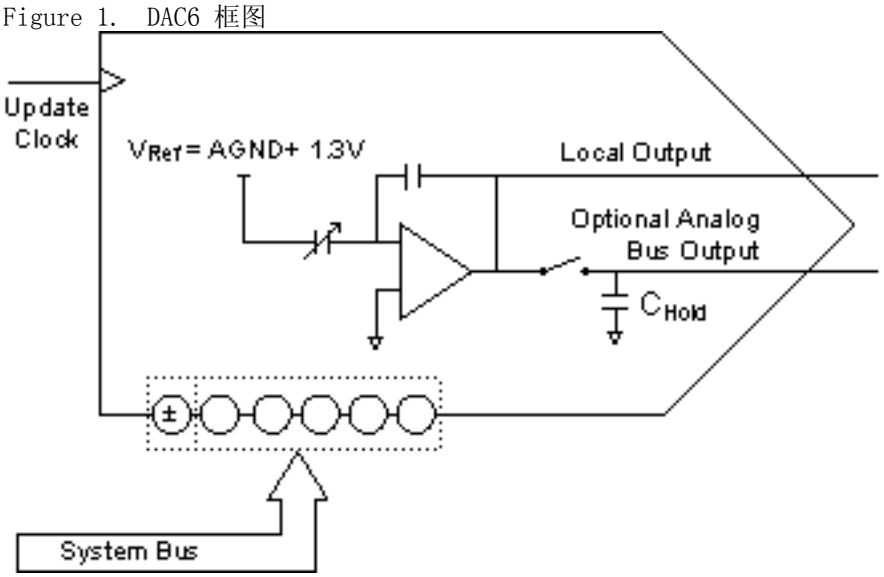
如需一个或多个使用此用户模块的完全配置的功能性示例工程，请转到 www.cypress.com/psocexampleprojects.

功能和概述

- 6-bit 分辨率
- 电压输出
- 二进制补码、偏移二进制和符号 / 大小输入数据格式
- 模拟总线和外部输出的采样和保持
- 更新速率达到 250 ksps

DAC6 用户模块将数字代码转换为输出电压。 DAC6 用户模块以高达每秒 250k 的更新速率将数字代码转换为输出电压。 应用程序编程接口 (API) 支持偏移二进制、符号与大小和二进制补码数据格式，拥有最大的灵活性。 偏移补偿用于将错误减到最少。

Figure 1. DAC6 框图

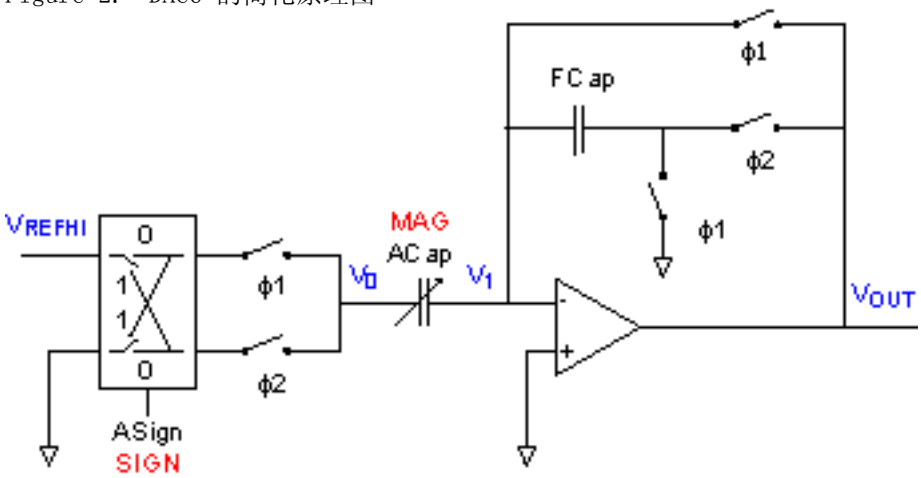


功能说明

DAC6 用户模块将数字代码转换为模拟输出电压。 数字代码显示为从 -31 到 +31 的二进制补码或符号与大小格式的数字。 也可以使用从 0 到 62 的偏移二进制数字显示输入代码。 这意味着输出电压的一步更改代表全量程输出范围的 1/63，而不是较为常见的 1/64。 在符号与大小格式下，输入代码 “-0” 会通过用户模块 API 转换为 “+0”。 输出电压范围根据针对系统级参数 REFMux 所选择的值，可以有多个选择。

在内部进行操作时基于符号与大小格式。 五个大小位设置了 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 使用 0 到 31 个单位范围内的值。 可由 ASign 位反相的参考电压在输出时根据 ACap 与反馈电容 FCap 之间的比率标度，具有 32 个单位的标称电容。

Figure 2. DAC6 的简化原理图



符号位通过纵横开关与时钟信号 Φ_1 和 Φ_2 控制的开关对极性进行控制。 这两个信号在周期内是相同的，在相位中则相反。 Φ_1 和 Φ_2 为 “ 负遮盖 ”。 即每个脉冲之间存在一个较短周期，其间二者均处于非活动状态。 这可以保证 Φ_1 和 Φ_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。

输入代码为正值时（符号位为 0），纵横开关在 Φ_1 为活动状态时将 V_{REFHI} 连接到 ACap，为 ACap 充电，直至达到参考电压减去 AGND 后的值。 当 Φ_1 转为非活动状态， Φ_2 为活动状态时，ACap 的输入侧从 V_{REFHI} 切换为 AGND，并有效反转其相对于 AGND 的含义。 由于电荷在 ACap 和 FCap 之间分配，运算放大器将提供相反电荷。

因此，对于正向输入代码，将发生两次电压反转。 当 ACap 的源终端切换为 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。 主要区别在于 V_{REFHI} 直接在 Φ_2 期间应用，而不是 Φ_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 控制的开关会对运算放大器进行配置，将其作为 Φ_1 期间的单位增益跟随器。 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，因此其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。

在每个更新周期中，V_{out} 在运算放大器偏移电压 （于 Φ_1 期间设定）与所需电压 （于 Φ_2 期间设定）之间转换；直接结果是出现偏移补偿。 降低高精度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。 Vout 在 Φ_2 后半阶段为负载和保持电容（DAC6 框图中的 CHold）充电。 在该周期结束时，CHold 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。

Equation 1

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

如果 REFMux 参数由器件编辑器配置为 2*BandGap ± BandGap，则应用以下公式。

示例

如果为 DAC 输入代码指定的值为 16，则得出的输出电压可能为 3.25V，如公式 2 所示：

Equation 2

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

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。

直流和交流电气特性

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 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 DAC6 直流和交流电气特性，PSoC 器件的 CY8C29/27/24/22xxx 系列

参数	典型值	极限值	单位	条件和注释
分辨率	--	6	位	
线性度				
DNL	0.09	--	LSB	
INL	0.07	--	LSB	
单调	是	--		
增益误差				
包括参考增益误差	3.4	--	%FSR	
不含参考增益误差 [3]	0.45	--	%FSR	
V_{OS}，偏移电压	±7.5	--	mV	
输出噪声	4.6	--	mV rms	0 到 300 kHz
f_{clock}，模拟列时钟 [1]				

参数	典型值	极限值	单位	条件和注释
低功耗	0.128 到 0.5	--	MHz	
中等功耗	0.128 到 2.0	--	MHz	
高功耗	0.128 到 3.2	--	MHz	
工作电流 [2]				
低功耗	155	--	µA	
中等功耗	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 T_A = 25° C，Vdd = 3.3V。并且，f_{clock} = 125 kHz，外部 AGND 1.50V，外部 V_{Ref} 0.8V，REFPWR = HIGH，SCPOWER = ON，PSoC 模块功耗为 “HIGH”。

Table 2. 3.3V DAC6 直流和交流电气特性，PSoC 器件的 CY8C29/27/24/22xxx 系列

参数	典型值	极限值	单位	条件和注释
分辨率	--	6	位	
线性度				
DNL	0.09	--	LSB	
INL	0.07	--	LSB	
单调	是	--		
增益误差				
包括参考增益误差	2.9	--	%FSR	
不含参考增益误差 [3]	0.3	--	%FSR	
V_{OS}，偏移电压	±7.5	--	mV	
输出噪声	2.1	--	mV rms	0 到 300 kHz
f_{clock}，模拟列时钟 [1]				
低功耗	0.128 到 0.5	--	MHz	
中等功耗	0.128 到 2.0	--	MHz	
高功耗	0.128 到 3.2	--	MHz	

参数	典型值	极限值	单位	条件和注释
工作电流[2]				
低功耗	150	--	µA	
中等功耗	560	--	µA	
高功耗	2150	--	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 T_A = 25° C，Vdd = 2.7V。并且，f_{clock} = 125 kHz，外部 AGND 1.50V，外部 V_{Ref} 0.8V，REFPWR = HIGH，SCPOWER = ON，PSoC 模块功耗为 “HIGH”。

Table 3. 2.7V DAC6 直流和交流电气特性，PSoC 器件的 CY8C29/27/24/22xxx 系列

参数	典型值	极限值	单位	条件和注释
分辨率	--	6	位	
线性度				
DNL	0.09	--	LSB	
INL	0.07	--	LSB	
单调	是	--		
增益误差				
包括参考增益误差	2.9	--	%FSR	
不含参考增益误差[3]	0.3	--	%FSR	
V_{OS}，偏移电压	±7.5	--	mV	
输出噪声	2.1	--	mV rms	0 到 300 kHz
f_{clock}，模拟列时钟[1]				
低功耗	0.128 到 0.5	--	MHz	
中等功耗	0.128 到 2.0	--	MHz	
高功耗	0.128 到 3.2	--	MHz	
工作电流[2]				
低功耗	150	--	µA	
中等功耗	560	--	µA	
高功耗	2150	--	µA	

电气特性注释

1. 为宽频带噪声中 3dB 增加指定的范围上限。< 1 LSB 下降的下限。 由 DAC 选择的模拟列时钟频率比控制更新周期的相位时钟频率快四倍。 请参阅以下的时序讨论内容。
2. 不包括所有模拟模块均通用的参考模块功耗 （请参阅 PSoC 系列数据手册）。
3. 参考增益误差测量方法是将 $V_{RefHigh}$ 外部参考与通过测试复用器并返回至引脚的 V_{RefLow} 进行比较。
4. 除非下表中另有说明，否则在以下条件下保证所有限值：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 4. 5.0V DAC6 直流和交流电气特性，PSoC 器件的 CY8C26/25xxx 系列

参数	典型值[1]	限值[2]	单位	条件和注释
分辨率	--	6	位	
线性度				
DNL	.02	.05	LSB	
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}，模拟列时钟[4]	--	0.128 到 1.33	MHz	
工作电流[5]				
低功耗	125	--	?A	
中等功耗	280	--	?A	
高功耗	780	1000	?A	

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

Table 5. 3.3V DAC6 直流和交流电气特性，PSoC 器件的 CY8C26/25xxx 系列

参数	典型值[1]	限值[2]	单位	条件和注释[5]
分辨率	--	6	位	
线性度				
DNL	.02	.04	LSB	
INL	.04	.09	LSB	
单调性	--	½	位	
增益误差	1.0	2.5	%FSR	
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}，模拟列时钟[4]	--	0.128 到 1.33	MHz	
工作电流[5]				
低功耗	100	--	?A	
中等功耗	250	--	?A	
高功耗	640	900	?A	

电气特性注释

1. 典型值等于统计平均值加 1 秒。
2. 限值通过测试或统计分析加以保证。
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. 为宽频带噪声中 3dB 增加指定的范围上限。< 1 LSB 下降的下限。 由 DAC 选择的模拟列时钟频率比控制更新周期的相位时钟频率快四倍。 请参阅以下的时序讨论内容。
5. PSoC 模块电流要求唯一的参考电流。

放置

DAC6 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 DAC6 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，DAC6 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

要创建 DAC6 实例，请选择器件编辑器中的用户模块 （如果需要可重新命名），并将其映射到该器件的任意开关电容 PSoC 模块中。 如果信号在片外驱动并与列时钟资源上的其他用户模块相互依存，放置时需注意模拟列输出总线是否可用。 放置后，该用户模块会象征性地将此模块命名为 “DAC”，并显示其参数。

数据格式 (DataFormat)

DAC6 用户模块 API 可处理以下三种不同的数据格式：偏移二进制、二进制补码及符号与大小。 API 章节 （参见下文）的 WriteBlind 入口点部分对这些规范及每个规范相关值的范围进行了描述。

模拟总线 (AnalogBus)

DAC 模块将其输出传播到相邻的模拟 PSoC 模块。 选择其中一个模拟总线选项，将 DAC 输出通过一个模拟输出缓冲区与外界连接。 在特定列中选择该总线，为阵列顶部的 PSoC 模块提供额外的本地连接。 开关电容 PSoC 模块采用了一个采样和保持电路，可在 Φ_2 的后半阶段对 DAC 输出进行采样。 这样可以将外部输出与自动归零操作中产生的电压摆幅分离。

时钟相位 (ClockPhase)

此参数决定由列时钟分频器生成的相位时钟 （Φ_1 和 Φ_2）的角色，列时钟分频器将在接下来的时钟与时序各章节中讨论。 选择*正常*时，Φ_1 期间将发生自动归零循环，DAC 的输出在 Φ_2 期间有效。 将时钟相位 (ClockPhase) 设置为*交换*时，情况则相反。 当 DAC 连接到另一个在 Φ_1 期间对输入进行采样的外设时，您会用得上这一信息。

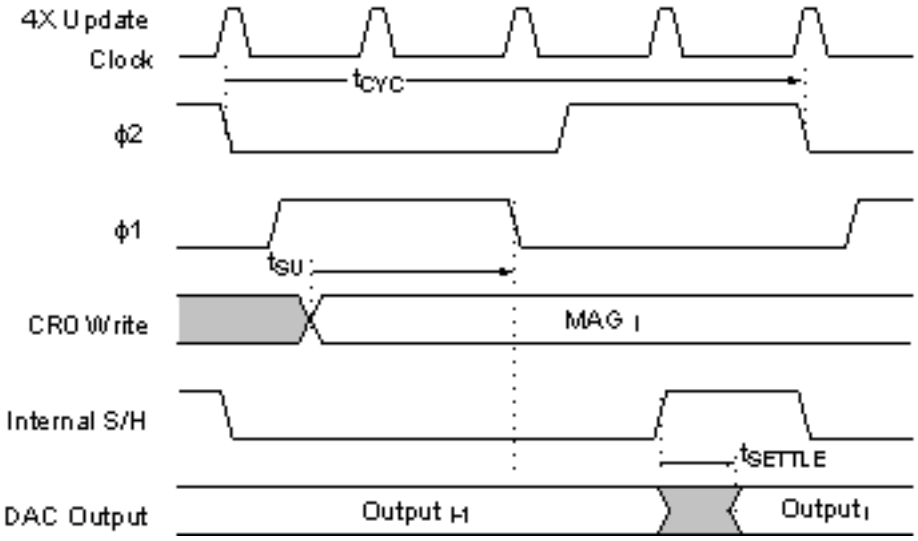
Note 将时钟相位 (ClockPhase) 设置为 “ 交换 ” 时将禁用模拟输出总线的采样和保持功能。 如果将 “ 模拟总线 ”(AnalogBus) 参数设置为 “ 使能 ”，总线输出将反映本地 PSoC 模块输出的情况，交替显示 Φ_1 期间的 AGND （加偏移电压）和 Φ_2 期间的所需输出。

模拟列时钟

不论 DAC 是否获得指令，指示其通过调用相应的 WriteBlind 和 WriteStall API 函数 “ 写入 ” 新值，其都会持续更新输出。 模拟列时钟复用器会选择用于生成相位时钟 Φ_1 和 Φ_2 的源时钟，相位时钟可控制此更新操作。 相位时钟发生器将列时钟四分频，生成 Φ_1 和 Φ_2，因此列时钟频率比实际的模拟输出更新速率要快四倍。 两个复用等级为包括所有数字模块和系统时钟分频器在内的列时钟提供了不同选择。 上面的 “ 电气特性 ” 章节对列时钟频率的上限和下限进行了说明。

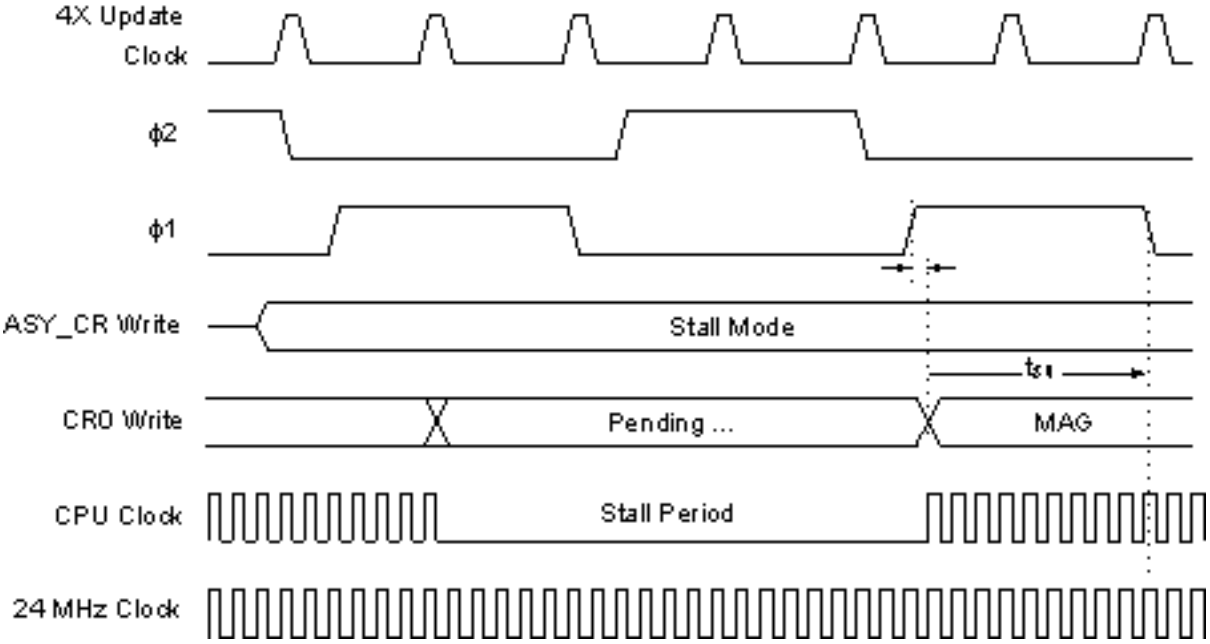
对于正输出电压 (Vout > AGND) 和正常相位， Φ_1 期间参考电压会存储在 A Cap 中，如上面的简化原理图所示。 为了对 A Cap 完全充电，对其值所作的任何更改都必须符合 Φ_1 下降沿的建立时间 t_{SU}。 此建立时间 （如下图所示）取决于参考功耗水平。 虽然建立时间未特性表征化，但硬件停止机理可用来保证符合建立时间的要求。 如果由于建立时间未满足要求而造成 A Cap 未完全充电，输出将会出错，直到到达整个 f1 周期的下一个相位时钟周期时才会得到更正。 f2 下降沿的类似建立时间会控制负输出电压 (Vout < AGND) 的行为。

Figure 3. V_{OUT} > AGND 的更新时序



对于一大类应用程序，允许出现瞬时 （一个更新周期）偏差。 其他应用程序可能有更为严格的要求。 可以利用硬件同步来控制对 A Cap 值进行更改的寄存器写入时序。 此操作在 WriteStall API 中由入口点直接支持。 调用时，基础硬件对向 PSoC 模块寄存器的写入操作加以识别，并冻结 CPU 时钟，保持写入完成状态，直至出现 Φ_1 的上升沿为止。 ASY_CR 寄存器会进行相应控制。

Figure 4. 硬件同步与 CPU 时钟停止



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 DAC 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。 CPU 周期的数量在同类停止期间的相应时间内丢失，相应周期可以使用以下关系进行计算。

Equation 3

$$CPU\,Cycles \le \frac{F_{CPU}}{F_{\phi_1}} = \frac{4 \times F_{CPU}}{F_{ColClock}}$$

要将停止期间丢失的 CPU 周期的数量减至最少，无疑应以实际最高频率运行列时钟。 由于额外的输出周期只重复之前的输出周期，所以，相比对输出电压进行实际更改所需的频率，这一频率要高得多。 较快的列时钟也会最小化从输出函数调用到输出更改时的延迟。

不过，对列时钟频率存在实际的限制。 由于 DAC 必须从 AGND 转化为每个相位时钟周期的输出电压，因此，列时钟的频率被限定为允许设定输出的大小。 当使用模拟输出总线的采样和保持功能时，运算放大器输出将在 Φ2 的后半阶段的采样窗口中驱动总线。 如果列时钟过快，导致运算放大器仍在转化并设定输出电压，则会观察到运算放大器输出信号的噪声。 极端情况下，在采样窗口关闭前该输出仅部分转化为最终输出电压。 这时可观察到输出中出现的严重非线性增益压缩，在输出范围全量程末端最为明显。 模拟列时钟可防止此现象的发生，其上限在上面的 “ 电气特性 ” 表中给出。

应用程序编程接口

应用程序编程接口 (API) 例程作为用户模块的一部分提供，从而使设计人员能够采用更高级的方式处理模块。 本部分具体指明了每个函数的接口以及由 “include” 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 寄存器的值可能通过调用 API 函数发生更改。 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。 选择这种 “ 寄存器易失 ” 策略是为了提高效率，并且从 PsoC Designer 的 1.0 版本起已开始使用。 C 编译器自动遵循此要求。 汇编语言编程人员也必须确保自己的代码遵守这一策略。 虽然有些用户模块的 API 函数可能保留 A 和 X 不变，但并不保证在未来也将如此。

所提供的入口点用来初始化 DAC6 用户模块、写入更新值，以及禁用用户模块。

DAC6_Start

说明：

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

C 原型：

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

汇编程序：

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

参数：

bPowerSetting： 一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。

符号名称	值
DAC6_OFF	0
DAC6_LOWPOWER	1
DAC6_MEDPOWER	2
DAC6_FULLPOWER	3

返回值：

无

副作用：

DAC 输出将被驱动。 默认情况下，初始值为 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。 用此函数会更改 A 和 X 寄存器。

DAC6_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  DAC6_SetPower(BYTE bPowerSetting)
```

汇编程序：

```
mov   A, bPowerSetting
lcall  DAC6_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：

无

副作用：

DAC 输出将被驱动。 默认情况下，初始值为 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。 用此函数会更改 A 和 X 寄存器。

DAC6_WriteBlind

说明：

立即将输出电压更新为指定值。

C 原型：

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

汇编程序：

```
mov   A, cOutputValue
lcall  DAC6_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 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
符号与大小 (SignAndMagnitude)	-31	31

“ 二进制补码 ”(TwosComplement) 与 “ 偏移二进制 ”(OffsetBinary) 使用微控制器的本地二进制补码格式。 偏移二进制值为正数，用 0 代表最低输出电压，62 代表最高输出电压。 在 “ 符号与大小 ”(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：

无

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。 用此函数会更改 A 和 X 寄存器。

注： 当您选择 OFFSET_BINARY 时，超出范围的输入值将自动转换成 API 内的二进制补码数据。 这表明，超过最大偏移二进制容许值的超范围值将转换为一个较小的正输出值 （接近 Agnd）。

DAC6_WriteStall

说明：

Φ_1 开始前可能会停止微处理器，然后更新输出电压到指定值。 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  DAC6_WriteStall (CHAR cOutputValue)
```

汇编程序：

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

参数：

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

返回值：

无

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。 用此函数会更改 A 和 X 寄存器。

DAC6_Stop

说明：

断开用户模块的电源。

C 原型：

```
void  DAC6_Stop(void)
```

汇编：

```
lcall  DAC6_Stop
```

参数：

无

返回值：

无

副作用：

将不会驱动输出。 用此函数会更改 A 和 X 寄存器。

固件源代码示例

以下示例代码创建了一个周期性下降的锯齿波。

```
//-------------------------------------------------------------------------
// 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
    }
}
```

以下汇编示例代码与 C 示例代码具有相同功能：

```
;-------------------------------------------------------------------------
; 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
```

配置寄存器

API 为 DAC6 用户模块提供了完整的接口。 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 DAC6 开关电容 DAC 模块。

Table 6. 模块 DAC ASAxxCR0 或 ASBxxCR0: 寄存器 CR0

位	7	6	5	4	3	2	1	0
值	1	0	符号与大小					

在执行复位与重新配置后 “ 符号与大小 ” 会设置为中等量程 (AGND)。 可通过在 API 中调用 “ 写入 ” 进行修改。

Table 7. 模块 DAC ASAxxCR1 或 ASBxxCR1: 寄存器 CR1

位	7	6	5	4	3	2	1	0
值	0	1	0	0	0	0	0	0

Table 8. 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 模块是否会驱动总线。 此位域的值取决于在器件编辑器的用户模块放置模式下对同名参数所作的选择。

Table 9. 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	电源	

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 10. 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。