6 到 14 位 Delta Sigma ADC 数据手册 V 1.30

Copyright © 2009-2014 Cypress Semiconductor Corporation. All Rights Reserved.

调制器订单	抽取速率	分辨率	采样率（CLK = 2 MHz）	采样率（CLK = 8 MHz）	数量抽取器	SC 模块	闪存	RAM	通道（I/O 引脚）
CY8C28x45、 CY8C28x43、 CY8C28x52、 CY8C28x33、 CY8C28x23									
1	32	6	15625.0	62500.0	2	2	148	3	2
1	64	7.5	7812.5	31250.0	2	2	156	3	2
1	128	9	3906.3	15625.0	2	2	185	5	2
1	256	10.5	1953.1	7812.5	2	2	185	5	2
2	32	8	15625.0	62500.0	2	2	187	5	2
2	64	10	7812.5	31250.0	2	2	216	7	2
2	128	12	3906.3	15625.0	2	2	216	7	2
2	256	14	1953.1	7812.5	2	2	216	7	2
1	32	6	15625.0	62500.0	3	6	180	6	3
1	64	7.5	7812.5	31250.0	3	6	192	6	3
1	128	9	3906.3	15625.0	3	6	234	9	3
1	256	10.5	1953.1	7812.5	3	6	234	9	3
2	32	8	15625.0	62500.0	3	6	215	6	3
2	64	10	7812.5	31250.0	3	6	257	9	3
2	128	12	3906.3	15625.0	3	6	257	9	3
2	256	14	1953.1	7812.5	3	6	257	9	3
1	32	6	15625.0	62500.0	4	8	200	7	4
1	64	7.5	7812.5	31250.0	4	8	216	7	4
1	128	9	3906.3	15625.0	4	8	271	11	4
1	256	10.5	1953.1	7812.5	4	8	271	11	4
2	32	8	15625.0	62500.0	4	8	243	7	4
2	64	10	7812.5	31250.0	4	8	298	11	4
2	128	12	3906.3	15625.0	4	8	298	11	4
2	256	14	1953.1	7812.5	4	8	298	11	4

有关其他转换器的信息，请参见应用手册 “ 模拟 — 模数转换器选择 ” AN2239。

特性和概述

- 6 到 14 位分辨率
- 2 到 4 个通道同步采样
- 数据为无符号或有符号 2 的补码格式
- 在 6 位分辨率情况下，最大采样率为 65,500 sps，在 14 位分辨率情况下，最大采样率为 7812 sps
- 在硬件中完全实现了 $Sinc^2$ 滤波器，并降低了 CPU 开销和抗锯齿要求
- 用户可以选择 1 阶或 2 阶调制器，以提高信噪比
- 输入范围由内部和外部参考选项定义
- 不需要数字模块
- 通过配置向导，可以轻松地选择彼此全部同步的 2、 3 或 4 通道 delta-sigma ADC 测量
- 抽取滤波器的内部定时器不允许使用数字模块

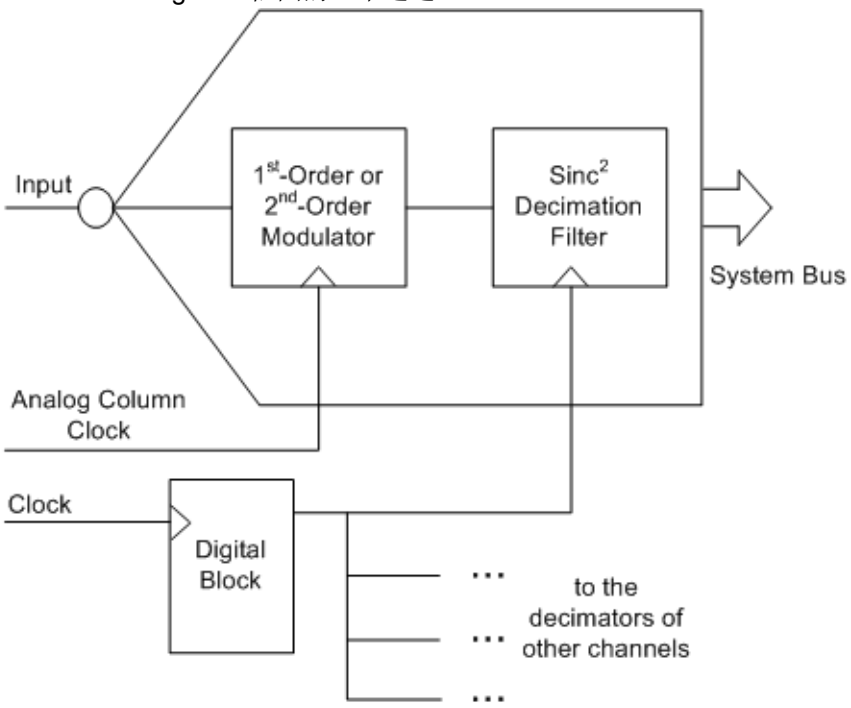
DelSigMulti 用户模块是积分转换器，需要 32 到 256 个积分周期，以生成单一个输出采样。更改复用输入会使得更改后的前两个采样失效。此 DelSigMulti 用户模块最多支持 4 通道同时同步 delta-sigma ADC 采样。

通过配置向导，可以轻松地选择每个通道使用的模拟模块数和每个通道的抽取滤波器过采样率。

在放置模块之前，请读取 “ 参数 ” 一节。

注意： 如果仅需要 1 个通道或需要多个不同步通道，则应当使用常规 “DelSig”、“DelSigPlus” 用户模块。

图 1. DelSigMulti 框图的一个通道



功能说明

如图 1 所示， DelSigMulti 用户模块由三个主要功能组成：

- 调制器
- Sinc^2 抽取滤波器
- 定时发生器

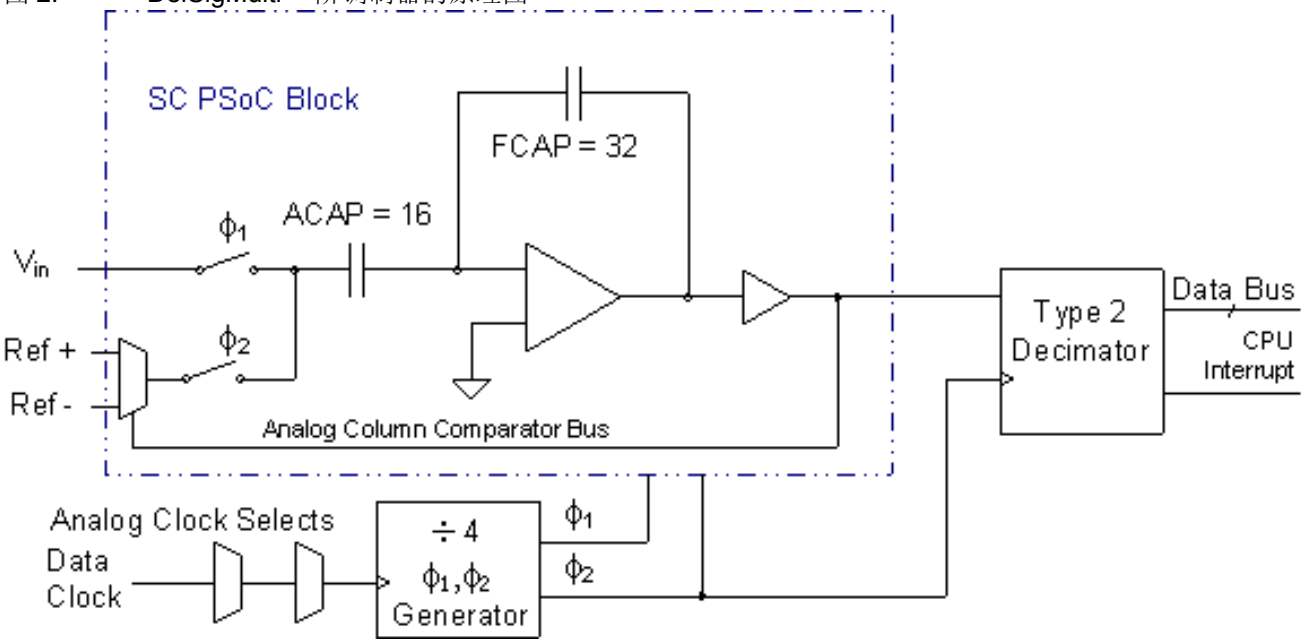
每个组件均提供了一些选项，可以通过调整这些选项对给定应用情况的性能与资源利用间合理的平衡。

调制器

调制器是 1 位过采样电路，它以所产生的 1 和 0 的密度形式表示输入电压。调制器输出通过低通抽取滤波器降低到最终采样率，该低通抽取滤波器会将多个 1 位样本转换为具有较高分辨率的样本。通常，抽取速率越高 （即过采样率越高），则分辨率结果越高，但是其他因素 （例如调制器的阶数）也会影响分辨率结果。

Delta-Sigma 转换器的主要优点是调制器可提供 “ 噪声整形 ”。通常，信号采样中固有的量化噪声是一种大致均匀分布的噪声 （白噪声），其频率介于 “DC” 与采样频率一半 （即奈奎斯特频率）之间。简单而言， delta-sigma 调制器会将某些量化噪声从较低频率转换为较高频率，之后这些频率会由抽取滤波器进行衰减。二阶调制器需要两个开关电容模拟 PSoC 模块，它对噪声整形的效果要好于仅需要一个模拟 PSoC 模块的一阶调制器。在最高抽取速率为 256X 时，与一阶调制器相比，二阶调制器将有效分辨率提高了 3.5 位。

图 2. DelSigMulti 一阶调制器的原理图



模拟模块配置为积分器。比较器的输出极性对参考复用器进行配置，以便在输入中增减参考电压，并置入积分器中。此参考控制尝试将积分器输出拉回零。一位比较器输出也会馈入到 sinc^2 抽取滤波器中。

请注意：1 位过采样率是由四分发生器决定的，四分发生器为生成控制开关电容 （SC） PSoC 模块的 φ_1 和 φ_2 时钟。确定输出速率的方法是：将数据除以 4，得到 1 位过采样率，然后再将该结果除以抽取速率得到最终采样率。

公式 1

$$SampleRate = \frac{DataClockFrequency}{4 \times DecimationRate} \text{ samples per second}$$

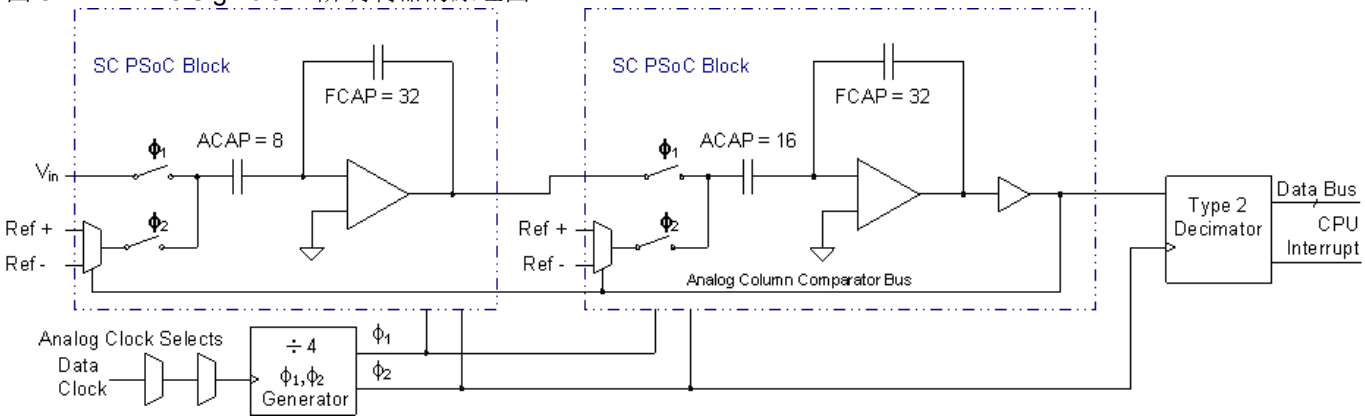
下面的规格表中给出了可以使用的最高数据时钟频率。对于 8 MHz 的数据时钟和 256 的抽取速率，采样率为：

公式 2

$$8 \times 10^6 / (4 \times 256) = 7812.5 sps$$

二阶调制器的构造方法是：将一阶调制器的模拟输出馈送到相似的 PSoC 模块中，并修改反馈排列，以便将第二个模块的 1 位比较器输出反馈给这两个模块，如下图所示：

图 3. DelSigMulti 二阶调制器的原理图



由于模拟比较器总线在模拟 PSoC 模块阵列的列中垂直运行，二阶调制器的模块必须叠加放置。

DelSigMulti 的范围是通过 $\pm V_{Ref}$ 建立的。您可以在 PSoC Designer “ 全局资源 ” 窗口中设置 V_{Ref}。对于固定量程，将 V_{Ref} 设置为 $\pm V_{Bandgap}$ ；对于 CY8C29x66 系列 PSoC 器件，设置为 ±1.6 $V_{Bandgap}$。对于可调整量程，将 V_{Ref} 设置为 ±Port 2[6]。为了提供比例式量程，将 V_{Ref} 设置为 $\pm V_{DD}/2$。下面的 1 表中，列出了完整选项：

表 1. 针对 Ref Mux 全局参数设置的输入电压范围

RefMux 设置	V_{DD} = 5 V	V_{DD} = 3.3 V
（$V_{DD}/2$）± 带隙	1.2 < V_{in} < 3.8	0.35 < V_{in} < 2.95
（$V_{DD}/2$）± （Vdd/2）	0 < V_{in} < 5	0 < V_{in} < 3.3
带隙 ± 带隙	0 < V_{in} < 2.6	0 < V_{in} < 2.6
（1.6* 带隙）± （1.6* 带隙）	0 < V_{in} < 4.16	NA

RefMux 设置	V_{DD} = 5 V	V_{DD} = 3.3 V
（2* 带隙）± 带隙	$1.3 < V_{in} < 3.9$	NA
（2* 带隙）± P2[6]	$(2.6 - V_{P2[6]}) < V_{in} < (2.6 + V_{P2[6]})$	NA
P2[4] ± 带隙	$(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$	$(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$
P2[4] ± P2[6]	$(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$	$(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$

Sinc^2 抽取滤波器

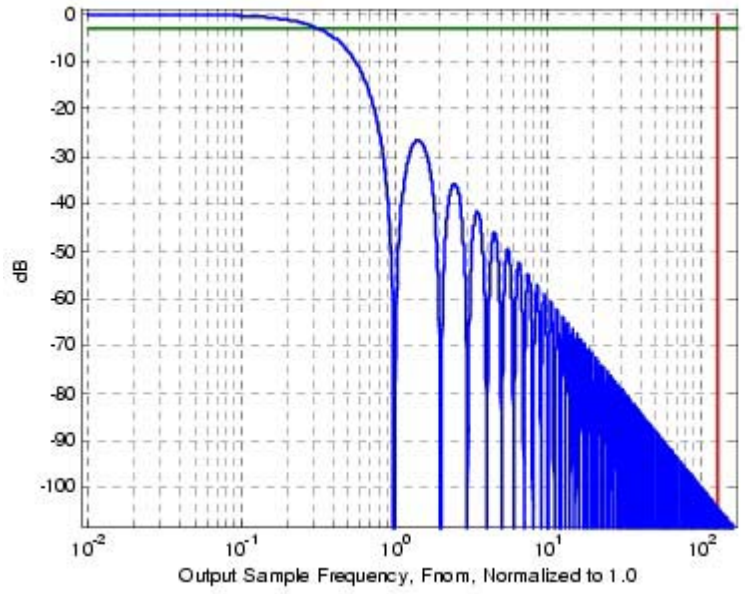
抽取过滤器的响应由下列 z 域关系提供：

公式 3

$$H(z) = \left[\frac{1 - z^{-n}}{1 - z^{-1}}\right]^2, \text{ where n is the decimation level.}$$

本章节绘制的频率域传输函数将频率标准化，使输出采样率 （F_{nom}）等于 1.0。 -3 dB 点出现在紧靠 $0.318 \times F_{nom}$ 上方，函数的零点出现在 F_{nom} 的每个整数倍数的位置。由于 1 位采样率比额定输出速率高 32 到 256， Nyquist 限制比 F_{nom} 高 4 到 7 个八度，因而极大降低了对抗锯齿滤波器的要求。在图形右侧，抽取速率为 256 的 1 位 Nyquist 频率以粗垂直线显示。虽然有可能实现较高的抽取速率，但是由于存在器件本底噪声，它们几乎不会产生的影响。对于 14 位拓扑，即抽取速率为 256 的二阶调制器，分辨率受信噪比限制。要在 DC 或慢速移动信号测量中获取可重复的 14 位分辨率，需要计算多个输出采样值的平均值，或者应用更复杂的信号处理技术实现。

图 4. Sinc^2 抽取滤波器量级响应



与早先的 DELSIG8 和 DELSIG11 不同，此用户模块在硬件中完整实现传输函数的分子和分母。这需要改进的 “ 类型 2” 抽取滤波器。该滤波器既可用于一阶调制器拓扑，也可用于二阶调制器拓扑。抽取滤波器通过以 1 位采样率运行的双积分器实现传输函数的分母。分子由以额定输出采样率运行的双微分器 （第二差分运算符）实现。 DelSigMulti 用户模块使用的 CPU 开销和中断延迟被限制在大约 80 个周期或更少的范围内，以便从 I/O 空间中的抽取滤波器寄存器获得采样数据。类型 2 抽取滤波器实际上是为 n 比特转换器生成从 0 到 2^n-1 的无符号值。可以配置中断服务例程，以将此转换为从 -2^{n-1} 到 $+2^{n-1}$-1 的 2 的补码值。

表 2. Delta Sigma ADC 特性表

特性	Delta Sigma ADC			
	DELSIG8、DELSIG11	DelSig	DelSigPlus	DelSigMulti
分辨率	8、 11	6-14	6-14	6-14
数字模块	1	1-2	0	0
模拟模块	1-2	1-2	1-2	2-8
支持的器件	支持 CY8C24/27/29，不支持 CY8C24x94、CY8C28x45	CY8C24x94、CY8C29xxx、CY8C28x45	CY8C24x94、CY8C28x45	CY8C28x45
CPU 开销和中断延迟	高	低	低	低

时序发生器和要求

向模拟调制器提供 φ_1 和 φ_2 时钟的四分时钟发生器也会向抽取滤波器提供位时钟。对应于输出采样率的抽取因子由字时钟确定。字时钟由抽取滤波器内部定时器生成。

类型 2 抽取滤波器是 $Sinc^2$ 滤波器的全硬件版本。其体系结构使您可以选择将内部定时器用于抽取和中断目的。要计算有效分辨率，请使用下列等式：

单一调制器：(log2(抽取滤波器速率) - 1) × 1.5

双端调制器：(log2(抽取滤波器速率) - 1) × 2

数据格式位可以加权为有符号 （2s 补码输出）或无符号 （偏移二进制数据）数值。

表 3. 抽取滤波器数据输出移位

抽取速率	调制器类型	有效分辨率	移位
32	单端	6	4
32	双端	8	2
64	单端	8 （7.5）	4
64	双端	10	2

抽取速率	调制器类型	有效分辨率	移位
128	单端	9	5
128	双端	12	2
256	单端	11 （10.5）	5
256	双	14	2

直流和交流电的电气特性

下列各值表示预计的性能，它们是根据初始特性数据得到的。除非另有指定，否则条件为：T_A = -40、 25、 85 和 125 °C， V_{dd} = 5.0 V。

表 4. 电压为 5.0 V 时的结果汇总

参数	典型值	限制	单位	注意
8 位、 24 MHz CPU Clk、 1 MHz 数据时钟、高功耗				
增益	-2.6482	2	%FSR	
偏移	-47.0072	13	mV	
DNL	0.161	<1	LSB	
INL	0.27	–	LSB	
SNR	45.86	–	dB	
8 位、 24 MHz CPU Clk、 2 MHz 数据时钟、高功耗				
增益	-2.3168	2	%FSR	
偏移	-62.3507	13	mV	
DNL	0.069	<1	LSB	
INL	0.172	–	LSB	
SNR	45.86	–	dB	

下列各值表示预计的性能，它们是根据初始特性数据得到的。除非另有指定，否则条件为：T_A = -40、 25、 85 和 125 °C， V_{dd} = 3.3 V。

表 5. 电压为 3.3 V 时的结果汇总

参数	典型值	限制	单位	注意
8 位、 24 MHz CPU Clk、 1 MHz 数据时钟、高功耗				
增益	-2.7182	2	%FSR	
偏移	-40.1334	5	mV	
DNL	–	<1	LSB	
INL	–	–	LSB	
SNR	–	–	dB	
8 位、 24 MHz CPU Clk、 2 MHz 数据时钟、高功耗				
增益	-2.8219	2	%FSR	
偏移	-42.8073	5	mV	
DNL	0.064	<1	LSB	
INL	0.161	–	LSB	
SNR	46.02	–	dB	

放置

如果 DelSigMulti 用户模块是在工具栏中选择的或者是通过在选择器视图中双击其图标选择的，将打开一个选择窗口，该窗口给出了选择适当拓扑的指南。稍后可以通过右键单击位置视图中的用户模块并从上下文菜单中选择 “User Module Selection Options...” （用户模块选择选项 ...）。

一阶调制器的设计需要一个 PSoC 模拟模块。名为 “ADC” 的模拟模块可以放置在任何开关电容 PSoC 模块中。

二阶调制器的设计使用两个开关电容 PSoC 模块：ADC0 和 ADC1。由于连接它们的模拟比较器总线在模拟阵列的每列中垂直运行，开关电容 PSoC 模块必须垂直累加放置。

虽然模拟模块有多种放置方法，但是 DelSigMulti 仍只使用 PSoC 器件的唯一硬件抽取滤波器。抽取滤波器是在放置模拟模块时自动分配的；不需要附加操作。因此，给定配置中只能放置 DelSigMulti 用户模块的一个实例。重新通过动态配置，可以同时加载 （激活）多个配置，而不用对阻止两个 DelSigMulti 用户模块同时运行而执行的检查。如果发生这种情况，则两个实例似乎都可以工作；但是，只有一个最近加载的实例可以控制抽取过滤器。两个中断仍可以运行，但是可能会有干扰。

参数和资源

放置 DelSigMulti 实例后，必须配置下列某些参数才能正确执行操作：输入信号复用器选择、时钟相位和轮询选择。

DataFormat0/1/2/3

此参数可以采用无符号 （默认值）或有符号值。对于 n 位分辨率，无符号数据采用的值为从零到 2^n-1。符号数据值属于 -2^{n-1} 到 $+2^{n-1}$-1 的范围内。 0/1/2/3 代表所使用的 delta-sigma ADC 通道数量。

Clock Phase0/1/2/3

时钟相位的选择用于将一个模拟 PSoC 模块的输出与另一个模拟 PSoC 模块的输入同步。开关电容模拟 PSoC 模块使用两相时钟 （φ_1、 φ_2）来获取和传输信号。通常， DelSigMulti 的输入是在 φ_1 上采样的。这便产生一个问题：许多用户模块在 φ_1 期间将其输出自动调零，仅在 φ_2 期间给出有效输出。如果此类模块的输出馈送到 DelSigMulti 的输入，则 DelSigMulti 对中间值采样。通过选择时钟相位，可以交换相位，以便在 φ_2 中获取输入信号。 0/1/2/3 表示所使用的 delta-sigma ADC 通道的数量。

PosInput0/1/2/3

此参数确定单端输入的信号源，或差分输入的非反相输入。 0/1/2/3 表示所使用的 delta-sigma ADC 通道数量。

NegInput0/1/2/3 与 NegInputGain0/1/2/3

NegInput 选择了一个差分信号对的反相输入源。如果使用单端输入，则可以将此参数设置为任意合法值。通过将 NegInputGain 参数设置为 “ 断开连接 ” （零增益），可以断开单端输入与转换器的连接。

NegInputGain 调整了反相输入相对于同相输入的增益 （请参见上述 NegInput 参数）。对于单端输入，此参数应该采用 “ 断开连接 ” 值。对于差分输入， NegInputGain 可以设置为 1.000。如果需要，应用于反相输入的增益还可以按 1/16 增量在 0.0625 与 1.9375 之间相对于同相输入进行调整。 0/1/2/3 代表所使用的 elta-sigma ADC 通道数量。

中断生成控制

PSoC Designer 中选中**使能中断生成控制**复选框时，该选框有两个可选的额外参数。该选框位于 **Project > Settings > Chip Editor** 路径下。当多个外覆层和多个用户模块跨外覆层所共享的中断一起使用时，中断生成控制起着关键作用：

- InterruptAPI
- IntDispatchMode

InterruptAPI

InterruptAPI 参数允许有条件生成用户模块的中断处理程序和中断向量表入口。选择 “Enable” （使能）以生成中断处理程序和中断向量表入口。选择 “Disable” （禁用）以取消生成中断处理程序和中断向量表入口。

如果项目有多个覆盖层，其中一个模块资源由不同的覆盖层使用，则选择时一定要加以注意。选择以仅为实际需要它们的覆盖层生成中断，以节省代码空间。

IntDispatchMode

IntDispatchMode 参数用于指定中断请求的处理方式，这些中断在同一个模块但在不同外覆层中的多个用户模块所共享。当您选择 ActiveStatus 时，固件在为共享中断请求提供服务之前测试哪个覆盖层处于活动。每次共享的中断发出请求时，都会进行该测试操作。这会增加延迟，还会产生为共享中断请求提供服务的不确定过程，但是不需要使用任何 RAM。当您选择 OffsetPreCalc 时，仅当最初加载覆盖层时，固件才会计算共享中断请求的来源。这种计算可减少中断延迟，并产生为共享中断请求提供服务的确定过程，但会占用一个字节的 RAM 空间。

应用编程接口 （API）

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。本章节指定每个函数的接口和 “include” 文件给出的相关常量。

每次放置用户模块时，都会为其分配一个实例名称。在默认情况下， PSoC Designer 向给定项目中此用户模块的第一个实例分配 DelSigMulti_1。可将该值更改为符合标识符语法规则的任意唯一值。分配的实例名称成为每个全局函数名称、变量和常量符号的前缀。为简便起见，在以下说明中将实例名称缩写为 DelSigMulti。

注意：

在这里，如同所有用户模块中的 API 一样，可通过调用 API 函数，进行更改 A 和 X 寄存器的值。如果调用后需要使用 A 和 X 的值，则调用函数要保留调用前的 A 和 X 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 版本起已强制使用。 C 编译器会自动遵循该要求。汇编语言编程员也要确保其代码遵循该策略。虽然一些用户模块 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，调用程序需要保留 CUR_PP、 IDX_PP、 MVR_PP 以及 MVW_PP 寄存器中的所有值。尽管目前尚未修改某些寄存器，但无法保证在将来的版本中也会如此。

DelSigMulti_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。所有通道都是在相同电源级别下配置的。

C 原型：

```
void  DelSigMulti_Start(BYTE bfPowerSetting)
```

汇编：

```
mov   A, bfPowerSetting
lcall  DelSigMulti_Start
```

参数：

bPowerSetting：一字节有四个 2 位字段，每个字段分别指定每个通道的功耗级别。在复位和配置之后，分配给 DelSigMulti 的模拟 PSoC 模块关闭电源。下表列出了 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	屏蔽
DelSigMulti_CH0_OFF	00h
DelSigMulti_CH0_LOWPOWER	01h
DelSigMulti_CH0_MEDPOWER	02h
DelSigMulti_CH0_HIGHPOWER	03h
DelSigMulti_CH1_OFF	00h
DelSigMulti_CH1_LOWPOWER	04h
DelSigMulti_CH1_MEDPOWER	08h
DelSigMulti_CH1_HIGHPOWER	0Ch
DelSigMulti_CH2_OFF	00h
DelSigMulti_CH2_LOWPOWER	10h
DelSigMulti_CH2_MEDPOWER	20h
DelSigMulti_CH2_HIGHPOWER	30h
DelSigMulti_CH3_OFF	00h
DelSigMulti_CH3_LOWPOWER	40h
DelSigMulti_CH3_MEDPOWER	80h
DelSigMulti_CH3_HIGHPOWER	C0h

返回值：

无

其他影响：

通过本次或下次执行该函数可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。

DelSigMulti_Stop

说明：

将开关电容 PSoC 模块的功耗级别设置为关闭。

C 原型：

```
void  DelSigMulti_Stop(void)
```

汇编：

```
lcall  DelSigMulti_Stop
```

参数：

无

返回值：

无

其他影响：

通过本次或下次执行该函数可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，调用 fastcall16 函数时，可通过调用函数将保留各个寄存器的值。

DelSigMulti_SetPower

说明：

设置 UM 的每个通道的开关电容 PSoC 模块的功耗级别。

C 原型：

```
void  DelSigMulti_SetPower(BYTE bPowerSetting)
```

汇编：

```
mov   A, bPowerSetting
lcall  DelSigMulti_SetPower
```

参数：

bPowerSetting：一字节有四个 2 位字段，每个字段分别指定每个通道的功耗级别。在复位和配置之后，分配给 DelSigMulti 的模拟 PSoC 模块关闭电源。下表列出了 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	屏蔽
DelSigMulti_CH0_OFF	00h
DelSigMulti_CH0_LOWPOWER	01h
DelSigMulti_CH0_MEDPOWER	02h
DelSigMulti_CH0_HIGHPOWER	03h
DelSigMulti_CH1_OFF	00h
DelSigMulti_CH1_LOWPOWER	04h
DelSigMulti_CH1_MEDPOWER	08h
DelSigMulti_CH1_HIGHPOWER	0Ch
DelSigMulti_CH2_OFF	00h
DelSigMulti_CH2_LOWPOWER	10h
DelSigMulti_CH2_MEDPOWER	20h
DelSigMulti_CH2_HIGHPOWER	30h

符号名称	屏蔽
DelSigMulti_CH3_OFF	00h
DelSigMulti_CH3_LOWPOWER	40h
DelSigMulti_CH3_MEDPOWER	80h
DelSigMulti_CH3_HIGHPOWER	C0h

返回值：

无

其他影响：

通过本次或下次执行该函数可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。

DelSigMulti_StartAD

说明：

激活此用户模块的中断并开始采样。

C 原型：

```
void  DelSigMulti_StartAD(void)
```

汇编

```
lcall  DelSigMulti_StartAD
```

参数：

无

返回值：

无

其他影响：

通过此函数的本次执行或下次执行可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。

DelSigMulti_StopAD

说明：

通过禁用中断，关闭 A/D。模拟电源仍提供给模拟模块。

C 原型：

```
void  DelSigMulti_StopAD(void)
```

汇编：

```
lcall  DelSigMulti_StopAD
```

参数：

无

返回值：

无

其他影响：

通过此函数的本次执行或下次执行可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。

DelSigMulti_fIsDataAvailable

说明：

检查新 ADC 采样数据是否已就绪。当 ADC 产生的结果已就绪时，进行设置标志。此 API 函数允许用户检查标志。为整个 UM 进行检查数据的可用性。所有通道都同步，因此所有数据同时是新数据或旧数据。因此，仅进行一项检查，该检查适用于所有通道的所有采样数据。用户必须使用另一个 API 函数清除此标志。

C 原型：

```
BYTE  DelSigMulti_fIsDataAvailable(void)
```

汇编：

```
lcall  DelSigMulti_fIsDataAvailable
cmp   A, 0
jz    .DataNotAvailable
```

参数：

无

返回值：

如果已转换每个通道的数据，且这些数据准备读取，则返回非零值。如果尚未转换每个通道的数据且这些数据未就绪，则返回零。

其他影响：

通过此函数的本次执行或下次执行可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。而当前，仅修改了 CUR_PP 页面指针寄存器。

DelSigMulti_GetAllDataClearFlag

说明：

此函数用于使用一个函数检索通道的数据。RAM 阵列指针传递到此函数中，此函数将 ADC 结果放入该 RAM 阵列。用户必须确保其 RAM 阵列大小适合 ADC 通道的数量和分辨率。此函数为 ADC 数据已就绪的信号清除标志。调用此函数之前，必须调用 DelSigMulti_fIsDataAvailable()，以确保数据采样已就绪。

C 原型：

```
void DelSigMulti_GetAllDataClearFlag(BYTE* pbRamBuffer)
```

汇编：

```
mov   A, >pbRamBuffer
mov   X, <pbRamBuffer
lcall  DelSigMulti_GetAllDataClearFlag
```

参数：

pbRamBuffer：此参数确定 ADC 采样数据复制到 RAM 位置。复制的数据为 ADC 结果。用户必须确保 RAM 阵列类型与 ADC 结果的数据类型相匹配。 ADC 通道数据放置在首先从通道 0 数据开始的用户阵列中，字节顺序为 MSB 优先。例如：如果 UM 有 3 个 10 位无符号数据的通道，则该阵列会按下列顺序将 6 个字节复制到用户阵列中：Ch0_Result_MSB、 Ch0_Result_LSB、 Ch1_Result_MSB、 Ch1_Result_LSB、 Ch2_Result_MSB、 Ch2_Result_LSB。如果用户拥有的阵列类型与 BYTE 类型是不同的，则用户应将传递给的函数的阵列指针强制传换成 BYTE 指针。

返回值：

无

其他影响：

通过此函数的本次执行或下次执行可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。当前，仅修改 CUR_PP 页面指针寄存器。

DelSigMulti_cGetData

DelSigMulti_iGetData

说明：

以有符号 8 位或 16 位 2 的补码格式返回转换的数据。请注意，用户模块 DataFormat 参数决定了基本表示。当基本表示是无符号数据时，调用有符号格式函数不会更改数据值。必须调用 DelSigMulti_fIsDataAvailable()，以确保数据采样已就绪。通道编号会传递给此函数。

C 原型：

```
CHAR DelSigMulti_cGetData(BYTE bChannelNumber) // use for 8-bit resolution or lower
INT  DelSigMulti_iGetData(BYTE bChannelNumber) // use for 9-bit resolution or higher
```

汇编：

```
mov   A, [bChannelNumber]
lcall  DelSigMulti_cGetData      ; Result will be in A
- or -
mov   A, [bChannelNumber]
lcall  DelSigMulti_iGetData      ; LSB will be in A, MSB in X upon return
```

参数：

bChannelNumber：此参数确定了被返回通道的数据。

返回值：

以 8 位或 16 位的 2 补码格式返回转换的数据采样。

其他影响：

通过此函数的本次执行或下次执行可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。当前，仅修改 CUR_PP 页面指针寄存器。

DelSigMulti_bGetData

DelSigMulti_wGetData

说明：

以 8 位或 16 位的无符号格式返回转换的数据。请注意，用户模块 DataFormat 参数决定了基本表示。当基本表示是有符号数据时，调用无符号格式函数不会更改数据值。必须调用 DelSigMulti_fIsDataAvailable()，以确保数据采样已就绪。通道编号会传递给此函数。

C 原型：

```
BYTE DelSigMulti_bGetData(BYTE bChannelNumber) // use for 8-bit resolution or lower
WORD DelSigMulti_wGetData(BYTE bChannelNumber) // use for 9-bit resolution or higher
```

汇编：

```
mov   A, [bChannelNumber]
lcall  DelSigMulti_bGetData       ; Result will be in A
- or -
mov   A, [bChannelNumber]
lcall  DelSigMulti_wGetData       ; LSB will be in A, MSB in X upon return
```

参数：

bChannelNumber：此参数确定了被返回通道的数据。

返回值：

根据该函数，以 8 位或 16 位的无符号格式返回转换的数据采样。

其他影响：

通过此函数的本次执行或下次执行可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。当前，仅修改 CUR_PP 页面指针寄存器。

DelSigMulti_ClearFlag

说明：

复位数据可用标志。

C 原型：

```
void  DelSigMulti_ClearFlag(void)
```

汇编：

```
lcall  DelSigMulti_ClearFlag
```

参数：

无

返回值：

无

其他影响：

通过此函数的本次执行或下次执行可以修改 A 和 X 寄存器在大型存储器模型中，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，当调用 fastcall16 函数时，调用函数将保留各个寄存器的值。当前，仅修改 CUR_PP 页面指针寄存器。

固件源代码示例

注意： 应将 DelSigMulti 用户模块设置为双通道，其中每个通道的分辨率不超过 8 位。

下面是汇编语言示例：

```
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

    ; two bytes are required for this example code
BUFF_SIZE: equ 2;

AREA userRAM(RAM,REL)
    ; reserve a 2-bytes area in RAM to load data in
RamBuffer: blk BUFF_SIZE

AREA text(ROM, REL)
export _main

_main:
   M8C_EnableGInt                  ; enable global interrupts
   mov   A,DelSigMulti_CH0_HIGHPOWER | DelSigMulti_CH1_HIGHPOWER        ; Establish
power setting...
   call  DelSigMulti_Start             ; and initialize
   call  DelSigMulti_StartAD           ; Commence sampling process
mainloop:
   call  DelSigMulti_fIsDataAvailable   ; Retrieve the status byte
   cmp   A, 0
   jz    mainloop                  ; spin lock until(data is Available)
   mov   A, >RamBuffer
   mov   X, <RamBuffer
   call  DelSigMulti_GetAllDataClearFlag   ; fastcall convention places ADC results in
RAM array defined by RamBuffer pointer
   call  ProcessSample             ; pass the sample to the dummy fcn
   jmp   mainloop

ProcessSample:
   ;...                            ; (do something useful with the data)
   ret
```

等效的 C 代码：

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

BYTE RamBuffer[2];

void main(void)
{
    M8C_EnableGInt;
        // start both channels of DelSigMulti ADC
   DelSigMulti_Start(DelSigMulti_CH0_HIGHPOWER|DelSigMulti_CH1_HIGHPOWER);
   DelSigMulti_StartAD();

    while(1)
    {       //  check if data sample is available
        if( DelSigMulti_fIsDataAvailable() )
        {
            DelSigMulti_GetAllDataClearFlag(RamBuffer); // copy ADC data to buffer
before processing
}
   }
}
```

配置寄存器

下面一节详述了此用户模块更改的寄存器。

模拟寄存器、一阶调制器

表 6. “ADC” 模拟开关电容 PSoC 模块使用的寄存器

寄存器	7	6	5	4	3	2	1	0
ADC_CH0/1/2/3_CR0	1	0	0	1	0	0	0	0
ADC_CH0/1/2/3_CR1	PosInput0/1/2/3			InvertingGain0/1/2/3				
ADC_CH0/1/2/3_CR2	0	1	0	0	0	0	0	0
ADC_CH0/1/2/3_CR3	1	1	1	0	NegInput0/1/2/3		功耗	

PosInput0/1/2/3 选择单端输入信号或差分输入信号的同相输入。 NegInput0/1/2/3 选择差分输入的反相输入。只要 InvertingGain0/1/2/3 字段设置为零，反相输入就会断开连接。通过 DelSigMulti_Start 和 DelSigMulti_SetPower API 函数设置功耗级别。 0/1/2/3 代表所使用的 delta-sigma ADC 通道的数量。

模拟寄存器、二阶调制器

表 7. “ADC0” 和 “ADC1” 模拟开关电容 PSoC 模块所使用的寄存器

寄存器	7	6	5	4	3	2	1	0
ADC0_CH0/1/2/3_CR0	1	0	0	0	1	0	0	0
ADC0_CH0/1/2/3_CR1	PosInput0/1/2/3			InvertingGain0/1/2/3				
ADC0_CH0/1/2/3_CR2	0	1	0	0	0	0	0	0
ADC0_CH0/1/2/3_CR3	1	1	1	0	NegInput0/1/2/3		功耗	
ADC1_CH0/1/2/3_CR0	1	0	0	1	0	0	0	0
ADC1_CH0/1/2/3_CR1	LinkToADC0			0	0	0	0	0
ADC1_CH0/1/2/3_CR2	0	0	0	0	0	0	0	0
ADC1_CH0/1/2/3_CR3	1	1	1	0	0	0	功耗	

PosInput0/1/2/3 选择单端输入信号或差分输入信号的同相输入。 NegInput0/1/2/3 选择差分输入的反相输入。只要 InvertingGain0/1/2/3 字段设置为零，反相输入就会断开连接。 LinktoADC0 由模块位置确定，它将 ADC0 模块的输出连接到 ADC1 PSoC 模块的 “A” 输入电容。 DelSigMulti_Start 和 DelSigMulti_SetPower API 函数设置了功耗级别。 0/1/2/3 代表所使用的 delta-sigma ADC 通道数量。

抽取滤波器控制寄存器

表 8. 抽取滤波器控制寄存器

位	7	6	5	4	3	2	1	0
DECx_DH	数据高字节							
DECx_DL	数据低字节							
DEC_CR0	ACC_IGEN				ICLKS	ACE_IGEN		DCLKS0
DEC_CR1	0	IDEC	ICLKS3	ICLKS2	ICLKS1	DCLKS3	DCLKS2	DCLKS1
DECx_CR0	POL	GOOO	GOOE	0		DATA_IN		
DEC_CR3	DEC1_EN	CLK_IN1			DEC0_EN	CLK_IN0		
DEC_CR4	DEC3_EN	CLK_IN3			DEC2_EN	CLK_IN2		
DEC_CR5	0				DSCLK			
DECx_CR	模式		数据输出移位		数据格式	抽取速率		

抽取滤波器是用于实现 Sinc2 滤波器的专用硬件。它由三个控制寄存器和两个数据输出寄存器组成。 DCol 选择连接哪个列比较器。 DCLKSEL 选择使用哪个数字模块来控制抽取滤波器定时。这两个参数都是在器件编辑器中设置的。 DEC_CR2 中的移位是根据抽取速率设置的，在 DEC_CR2 中也进行了指定，其目的是最大程度地减少软件中必须完成的对齐数据。

数字模块寄存器

表 9. 数字模块控制寄存器

位	7	6	5	4	3	2	1	0
DxCxxCR0								使能

此寄存器在使用时配置有附加数字模块，以便通过调用函数 DelSigMulti_StartAD 启动此 UM 的生成时钟。该寄存器通过调用函数 ADelSigMulti _StopAD 来停止生成时钟。

版本历史记录

版本	创作者	说明
1.2	DHA	增加了 DRC，以检查源时钟在数字源和模拟源中是否不同。 改进了 GetAllDataClearFlagAPI 实现。 改进了时钟选择 DRC。
1.2.b	DHA	在向导中添加了帮助文件。 更新了本用户模块数据手册中的代码示例。
1.30	MYKZ	1. 已经将模拟用户模块放置在非相邻的列中。 2. 增加了在 ADC 时钟超过 8 MHz 时的设计规则检查。

注意： PSoC Designer 版本 5.1 在所有用户模块数据手册中都介绍了 “ 版本历史记录 ”。本数据手册详细介绍当前和先前用户模块版本之间的区别。

Copyright © 2009-2014 赛普拉斯半导体公司。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品内嵌的电路外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任。也不会根据专利权或其他权利以明示或暗示的方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯产品不保证能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于合理预计会发生运行异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯将不批准将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

PSoC Designer™ 和 Programmable System-on-Chip™ 是赛普拉斯半导体公司的商标， PSoC® 是赛普拉斯半导体公司的注册商标。此处引用的所有其他商标或注册商标归其各自所有者所有。

所有源代码 （软件和 / 或固件）均归赛普拉斯半导体公司 （赛普拉斯）所有，并受全球专利法规 （美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和 / 或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定用途外，未经赛普拉斯的明确书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

免责声明：赛普拉斯不针对该材料提供任何类型的明示或暗示保证，包括 （但不仅限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不另行通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于合理预计可能发生运转异常和故障，并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统，则表示制造商将承担因此类使用而导致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用可能受适用于赛普拉斯软件许可协议的限制。