11-Bit Delta Sigma ADC 数据手册 DELSIG11 V 3.2

Copyright © 2002-2011 Cypress Semiconductor Corporation. All Rights Reserved.

资源	PSoC® 模块			API 内存 （字节）		引脚 （根据外部 I/O）
	数字	模拟 CT	模拟 SC	闪存	RAM	
CY8C29/27/24/22xxx, CY8C23x33, CY8CLED08/16, CY8C28x43						
二阶调制器	1	0	1	177	9	1
二阶调制器	1	0	2	200	9	1
CY8C26/25xxx						
二阶调制器	1	0	1	187	9	1

请参见 AN2239, ADC Selection Guide 了解其他转换器。

功能和概述

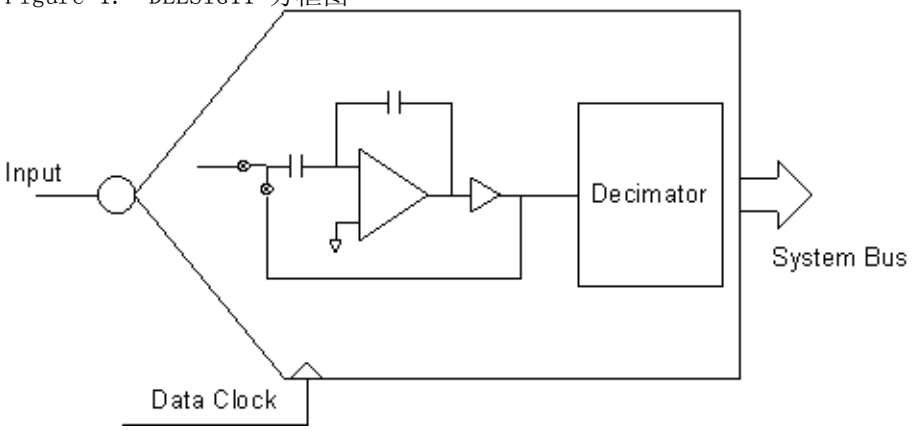
- 11-bit 分辨率
- 适合于 2 的补码的数据格式
- 采样率达 7.8 ksps
- 带有 $sinc^2$ 滤波器的 256X 过采样降低了消除信号混叠的要求
- 输入范围由内部和外部参考选项定义
- 内部或外部时钟

Note 如果此用户模块用于 CY8C29xxx 系列，它将额外消耗 6 毫安。 替代方法是使用 Delsig 用户模块。

DELSIG11 用户模块提供 11-bit 输出。 它基于用户选择的 AGND 为中心的 2.6V 全量程输入范围，前提是全局参数窗口中的参考选择设置为 +/- 带隙。 DELSIG11 支持的采样率范围为 125 sps 到 7.8 ksps，并提供 2 的补码输出。 采样率由数据时钟输入确定，但用户可以选择。 DELSIG11 生成的数据在收集数据的中断子程序中提供，或者通过由 DELSIG11 API 给出的轮询函数提供。

DELSIG11 是流水线积分转换器，需要 511 个积分周期才能生成一个输出采样。 如果此转换器具有复用输入，则两个样品必须先通过，第三个样品和后续样品才能有效。 请注意，需要在模块放置之前检查 “ 参数 ” 部分。

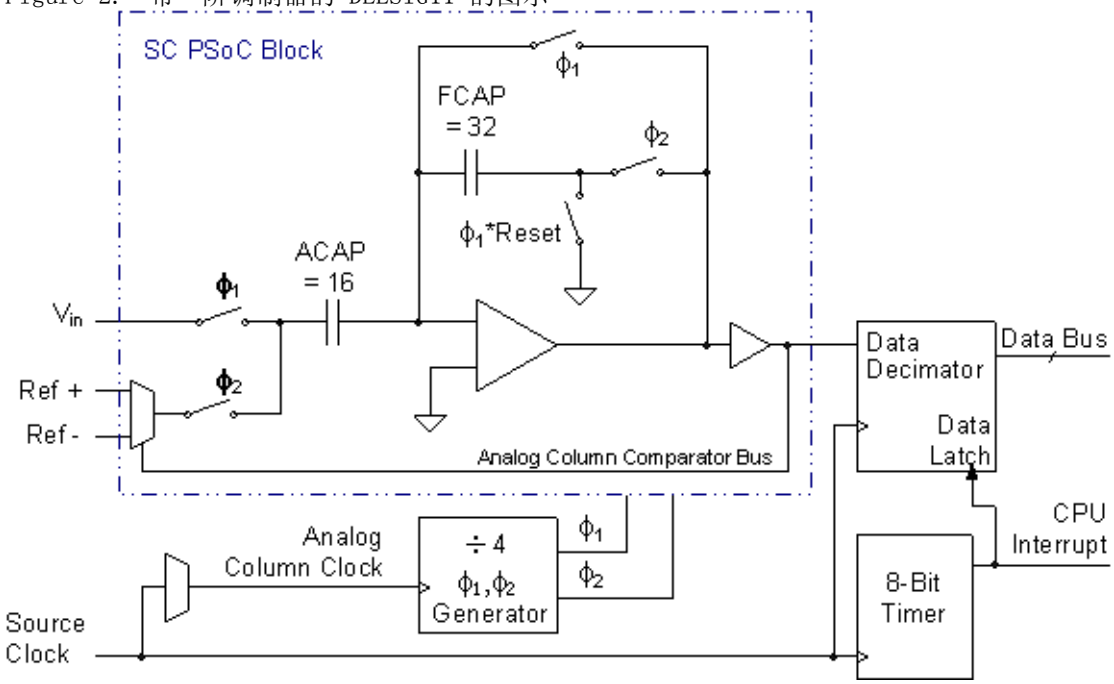
Figure 1. DELSIG11 方框图



功能说明

DELSIG11 提供了由一个模拟开关电容 PSoC 模块、一个数字 PSoC 模块和抽取滤波器构成的一阶调制器，如下图所示。

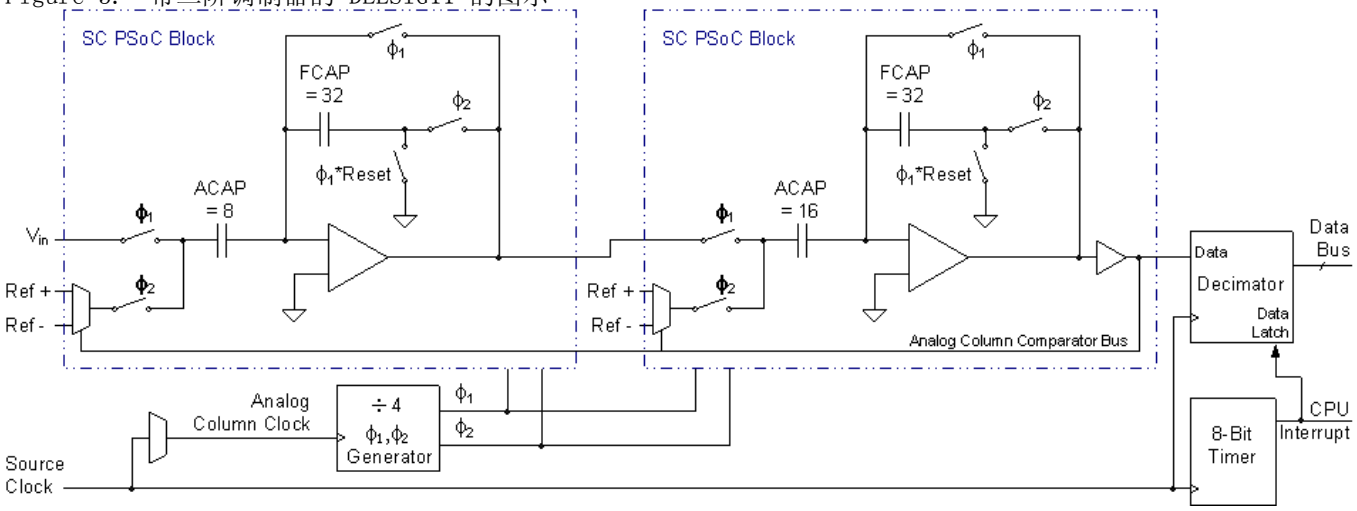
Figure 2. 带一阶调制器的 DELSIG11 的图示



[+] Feedback

二阶调制器可以使用另一个开关电容 PSoC 模块，在 CY8C29/27/24/22xxx, CY8C23x33, CY8CLED08/16, CY8C28x43 系列 PSoC 器件中构建。 它将更多量化噪声移到带外，提高了信噪比，从而改善了性能。 下图中显示的二阶电路使用模拟列比较器总线来调制参考选择。

Figure 3. 带二阶调制器的 DELSIG11 的图示



DELSIG11 的范围设置为 $\pm V_{Ref}$，其中 V_{Ref} 由用户在 PSoC Designer 的全局资源窗口中设置。 对于固定量程，V_{Ref} 设置为 $\pm V_{Bandgap}$；对于 CY8C29/27/24/22xxx, CY8C23x33, CY8CLED08/16, CY8C28x43 系列 PSoC 器件，设置为 $\pm 1.6\ V_{Bandgap}$。 对于可调整量程，V_{Ref} 设置为 ±Port 2[6]。 对于供电比率计量程，V_{Ref} 设置为 $\pm V_{DD}/2$。

模拟模块配置为积分器。 对参考控制进行配置，以便可以根据输出极性，从输入中增减参考电压，并置于积分器中。 此参考控制尝试将积分器输出拉回零。 单比特比较器输出馈送到用于实现抽取滤波器 $sinc^2$ 滤波器的电路中。 此滤波器的响应由下列 z 域关系给出。

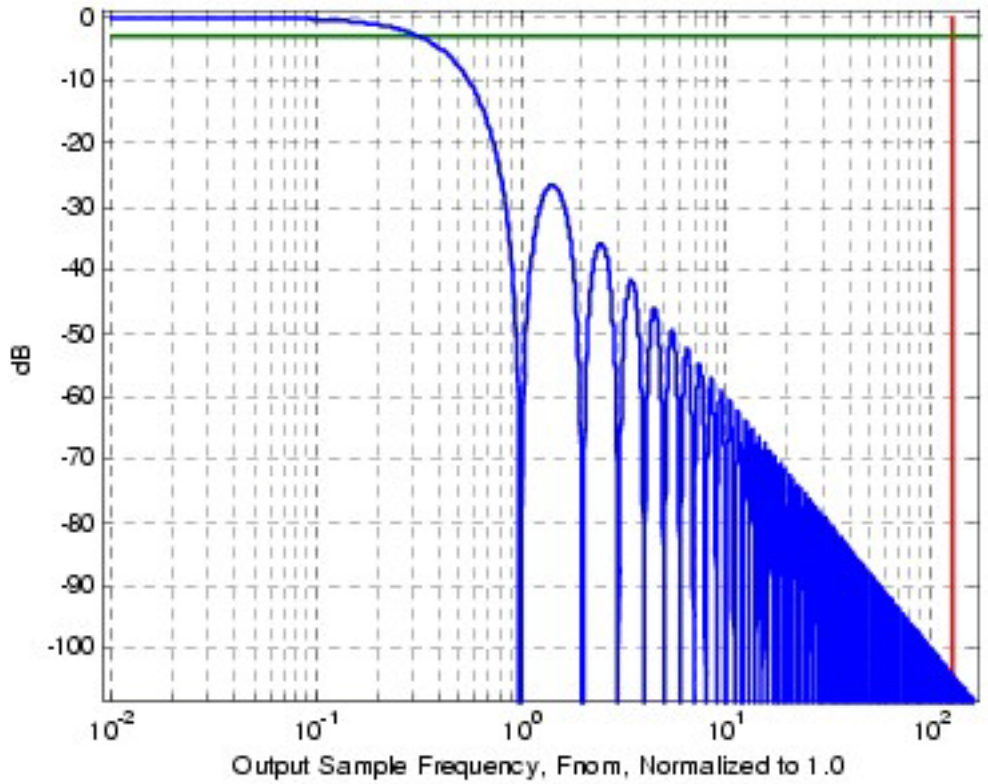
Equation 1

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

[+] Feedback

下面绘制的频率域量级图使频率标准化，以使 11-bit 采样率 F_{nom}=1.0。 -3 dB 点正好位于 0.318×F_{nom} 上方，函数的零点位于 F_{nom} 的每个整数倍数的位置。 由于 DELSIG11 的实际采样速率比额定输出速率快 256 倍，奈奎斯特限制比 F_{nom} 高 128 倍、7 个八度，这可以极大降低对防锯齿滤波器的要求。

Figure 4. 带 -3dB 点和奈奎斯特频率的 $Sinc^2$ 抽取滤波器量级响应



此滤波器是通过硬件和软件的组合实现的，既可用于一阶调制器拓扑，也可用于二阶调制器拓扑。 此滤波器的分母是在硬件中实现的双积分器。 它以数据速率运行。 分子是双微分器。 它以抽取速率计算，在软件中实现。 要将积分器函数作为 delta-sigma ADC，请使用下列数字资源：

- 一个定时器，允许适当积分周期数。
- 一个抽取滤波器，用于处理来自模拟模块的单比特输出流。

通常，构建 11-bit delta sigma ADC 转换器需要 8-bit 定时器。 必须按用来设置积分器时钟频率的四分之一来设置它的时钟。 对于此应用，按用于列时钟的时钟 （其中放置了积分器模块）来设置定时器的时钟。 这会导致定时器针对每个积分周期按 4 递增，因此需要 10-bit 定时器。

Note 当放置此模块时，必须使用与模拟和数字模块相同的源时钟来配置它。 不这样做会导致其运行不正常。

定时器设置为每 256 个计数生成一个中断。 在生成中断的同时，抽取滤波器中的数据锁存到输出寄存器中。 这些周期中的四个周期 （1024 个计数）用于计算中断子程序中的 A/D 值。

DELSIG11 的转换时间可以低到 128 µs，因此快速检索数据非常重要。 检索数据是通过将用户数据处理程序代码插入到汇编文件 *DELSIG11INT.asm* 中的中断子程序 DELSIG11_ADConversion_ISR 来实现的。 插入代码的位置被清晰标记。 此用户模块的 “ 示例固件源代码 ” 部分中显示了两种不同的数据处理技术。

[+] Feedback

下面显示了控制 11-bit Delta Sigma ADC 的转换等式。 该等式表明输入范围限制为 V_{ref}。 下面的示例说明了此等式的使用：

Equation 2

$$V_{in} = \frac{n - 1024}{1024} V_{ref}$$

示例 1

对于 1.3V V_{ref}，可以轻松根据数据就绪时从增量 ADC 读取的值来计算输入电压。 可以使用的等式为：

Equation 3

$$V_{in} = \frac{n - 1024}{1024} 1.3$$

计算结果以 AGND 为参考。 对于 ADC 数据值为 1500，测量的电压可以计算为 0。60V:

Equation 4

$$V_{in} = \frac{1500 - 1024}{1024} 1.3 = 0.60V$$

计算出的值是理想值，可能因系统噪声和芯片偏移而异。

要确定给定特定输入电压情况下期望的代码，可以重新排列该等式，以给出：

Equation 5

$$n = \frac{1024 \cdot V_{in}}{V_{ref}} + 1024$$

示例 2

对于 1.3V V_{ref}，我们可以轻松根据输入电压计算预计的 ADC 代码。 可以使用的等式为：

Equation 6

$$n = \frac{1024 \cdot V_{in}}{1.3} + 1024$$

对于低于 AGND 的输入电压 -1V，根据此计算，ADC 代码预计为 236.3：

Equation 7

$$n = \frac{1024 \cdot (-1)}{1.3} + 1024 = 236.3$$

[+] Feedback

直流和交流电气特性

下列值表示期望的性能，它们基于初始特性数据。 除非下面另外指定，否则条件为：T_A = 25° C，V_{dd} = 5.0V，功耗 = 高，运算放大器偏压 = 低，输出参考在 P2[6] 上 1.25 外部 Vref 的情况下在 P2[4] 上的 2.5V 外部模拟地。

Table 1. CY8C29/27/24/22xxx, CY8C23x33, CY8CLED08/16, CY8C28x43 系列 PSoC 器件的 5.0V 一阶调制器直流和交流电气特征

参数	典型值	限制	单位	条件和注释
输入				
输入电压范围	---	Vss 到 Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
输入电容[1]	3	---	pF	
输入电阻	1/(C*clk)	---	W	
分辨率	---	11	比特	
采样率	---	125 到 7,800	sps	
信噪比	66	---	dB	
直流准确性				
DNL	0.25	---	LSB	列时钟 2 MHz
INL	0.5	---	LSB	
偏移误差	5	---	mV	
增益误差				
包括参考增益误差	3.0	--	% FSR	
不包括增益误差[2]	0.1	--	% FSR	
工作电流				
低功耗	180	---	uA	
中等功耗	840	---	uA	
高功耗	3450	---	uA	
数据时钟	---	0.032 到 8.0	MHz	数字模块和模拟列时钟的输入

[+] Feedback

Table 2. CY8C29/27/24/22xxx, CY8C23x33, CY8CLED08/16, CY8C28x43 系列 PSoC 器件的 5.0V 一阶调制器直流和交流电气特征

参数	典型值	限制	单位	条件和注释
输入				
输入电压范围	---	Vss 到 Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
输入电容[1]	3	---	pF	
输入电阻	1/(C*clk)	---	W	
分辨率	---	11	比特	
采样率	---	125 到 7,800	sps	
信噪比	65	---	dB	
直流准确性				
DNL	1.4	---	LSB	列时钟 2 MHz
INL	1.2	---	LSB	
偏移误差	5	---	mV	
增益误差				
包括参考增益误差	2.0	--	% FSR	
不包括增益误差[2]	0.1	--	% FSR	
工作电流				
低功耗	50	---	uA	
中等功耗	500	---	uA	
高功耗	1900	---	uA	
数据时钟	---	0.032 到 8.0	MHz	数字模块和模拟列时钟的输入

[+] Feedback

下列值表示期望的性能，它们基于初始特性数据。 除非下面另外指定，否则条件为：T_A = 25° C，V_{dd} = 3.3V，功耗 = 高，运算放大器偏压 = 低，输出参考在 P2[6] 上 1.25 外部 Vref 的情况下在 P2[4] 上的 1.64V 外部模拟地

Table 3. CY8C29/27/24/22xxx, CY8C23x33, CY8CLED08/16, CY8C28x43 系列 PSoC 器件的 3.3V 一阶调制器直流和交流电气特性

参数	典型值	限制	单位	条件和注释
输入				
输入电压范围	---	Vss 到 Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
输入电容[1]	3	---	pF	
输入电阻	1/(C*clk)	---	W	
分辨率	---	11	比特	
采样率	---	125 到 7,800	sps	
信噪比	66	---	dB	
直流准确性				
DNL	0.25	---	LSB	列时钟 2 MHz
INL	0.5	---	LSB	
偏移误差	5	---	mV	
增益误差				
包括参考增益误差	3.0	--	% FSR	
不包括增益误差[2]	0.3	--	% FSR	
工作电流				
低功耗	130	---	uA	
中等功耗	840	---	uA	
高功耗	3370	---	uA	
数据时钟	---	0.032 到 8.0	MHz	数字模块和模拟列时钟的输入

[+] Feedback

Table 4. CY8C29/27/24/22xxx, CY8C23x33, CY8CLED08/16, CY8C28x43 系列 PSoC 器件的 3.3V 一阶调制器直流和交流电气特性

参数	典型值	限制	单位	条件和注释
输入				
输入电压范围	---	Vss 到 Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
输入电容[1]	3	---	pF	
输入电阻	1/(C*clk)	---	W	
分辨率	---	11	比特	
采样率	---	125 到 7,800	sps	
信噪比	65	---	dB	
直流准确性				
DNL	1.4	---	LSB	列时钟 2 MHz
INL	1.1	---	LSB	
偏移误差	5	---	mV	
增益误差				
包括参考增益误差	2.0	--	% FSR	
不包括增益误差[2]	0.3	--	% FSR	
工作电流				
低功耗	50	---	uA	
中等功耗	500	---	uA	
高功耗	1900	---	uA	
数据时钟	---	0.032 到 8.0	MHz	数字模块和模拟列时钟的输入

电气特性注释

1. 包括 I/O 引脚。
2. 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。

[+] Feedback

除非另有指定，否则所有限制应确保：TA = -40° C 到 +85° C，Vdd = 5.0V ±10%，功耗 = 高，运算放大器偏压 = 低，输出参考在 P2[6] 上 1.25 外部 Vref 的情况下在 P2[4] 上的 2.5V 外部模拟地。

Table 5. CY8C26/25xxx 系列 PSoC 器件的 5.0V DELSIG11 直流和交流电气特性

参数	典型值[1]	限制	单位	条件和注释
输入				
输入电压范围[2]	---	Vss 到 Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
输入电容[3]	0.8	---	pF	
输入电阻[4,5]	1/(C*clk)	---	W	
分辨率	---	11	比特	2 的补码
采样率	---	0.125 到 7.8[6]	ksps	每秒采样数
信噪比[7]	65	62	dB	在 7.8 ksps 条件下
直流准确性				
INL	0.5	1	LSB	
DNL	1.1	0.5	LSB	
偏移误差	14	73	mV	
增益误差	0.3	2.5	% FSR	不包括参考误差
工作电流				
低功耗	140	---	uA	sps = 487.5
中等功耗	330	---	uA	sps = 1950
高功耗	970	1150	uA	sps = 7800
数据时钟	---	0.032 到 8.0[6]	MHz	数字模块和模拟列时钟的输入

[+] Feedback

除非另有指定，否则所有限制应确保：T_A = -40° C 到 +85° C，V_{dd} = 3.0 到 3.6V，功耗 = 高，运算放大器偏压 = 低，输出参考在 P2[6] 上 1.25 外部 Vref 的情况下在 P2[4] 上的 1.64V 外部模拟地。

Table 6. CY8C26/25xxx 系列 PSoC 器件的 3.3V DELSIG11 直流和交流电气特性

参数	典型值[1]	限制	单位	条件和注释
输入				
输入电压范围[2]	---	Vss 到 Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
输入电容[3]	0.8	---	pF	
输入电阻[4,5]	1/(C*clk)	---	W	
分辨率	---	11	比特	2 的补码
采样率	---	0.125 到 7.8[6]	ksps	每秒采样数
信噪比[7]	65	62	dB	在 7.8 ksps 条件下
直流准确性				
INL	0.5	1	LSB	
DNL	1.4	0.5	LSB	
偏移误差	8	73	mV	
增益误差	1	2.5	% FSR	不包括参考误差
工作电流				
低功耗	90	---	uA	sps = 487.5
中等功耗	140	---	uA	sps = 1950
高功耗	430	820	uA	sps = 7800
数据时钟	---	0.032 到 8.0[6]	MHz	数字模块和模拟列时钟的输入

电气特性注释

1. 典型值表示 +25° C 时的参数标准。
2. 高于最大值的输入电压会生成最大正读数。 低于最小值的输入电压会生成最大负读数。
3. 仅限用户模块，不包括 I/O 引脚。
4. 输入电容或电阻仅在模拟模块的输入直接连接到引脚时适用。
5. C = 输入电容，clk = 数据时钟 （模拟列时钟）。
6. 数据时钟 = 采样率 * 1024。
7. 信噪比 = 全量程单一音频除以 $F_{sample}/2$ 积分的总噪声的功率比。

[+] Feedback

CY8C29/24xxx 典型性能

这些图形限制为最能表现控制错误的输入范围的子集。 不建议采取用户模块设置为 LOWPOWER 的操作。

Figure 5. 作为功耗水平的函数的典型 DNL （CY8C24xxx，3.3V，25° C）

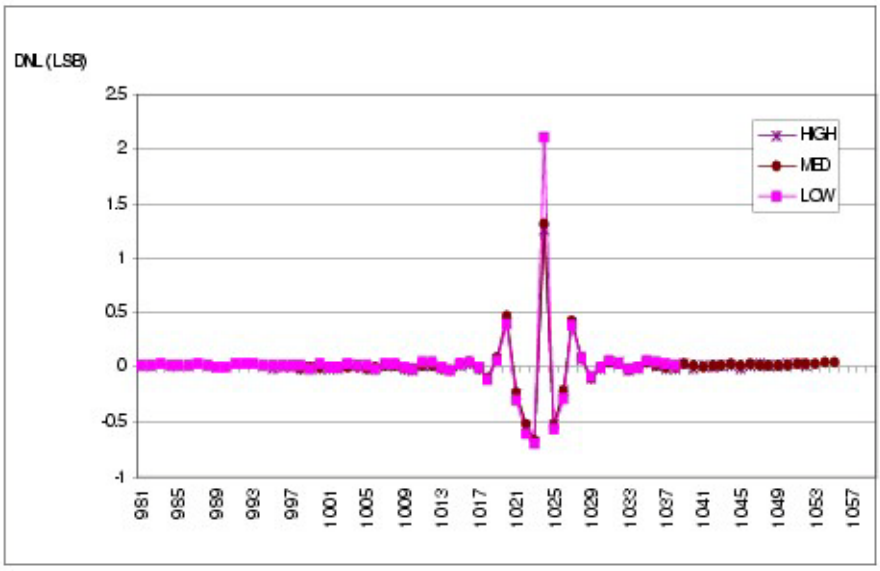
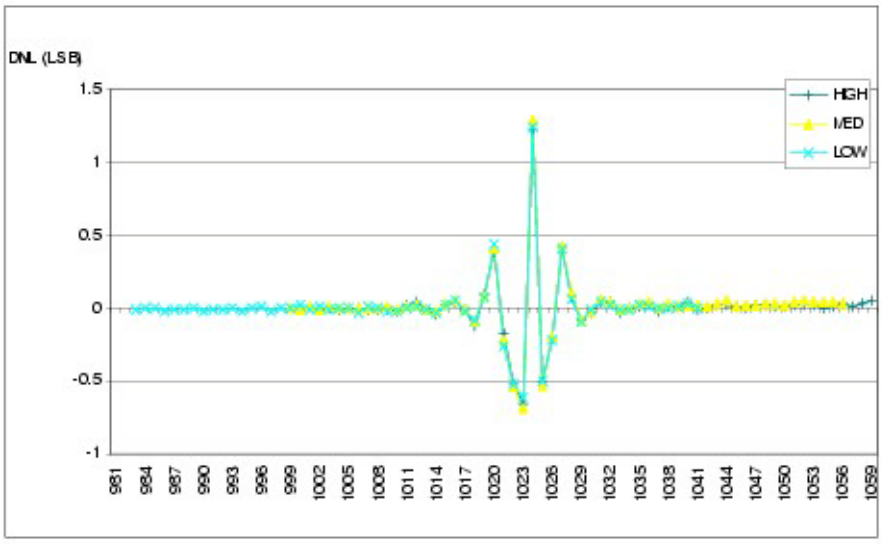


Figure 6. 作为功耗水平的函数的典型 DNL （一阶，CY8C24xxx，5.0V，25° C）



[+] Feedback

Figure 7. 作为功耗水平的函数的典型 DNL （二阶，CY8C24xxx，3.3V，25° C）

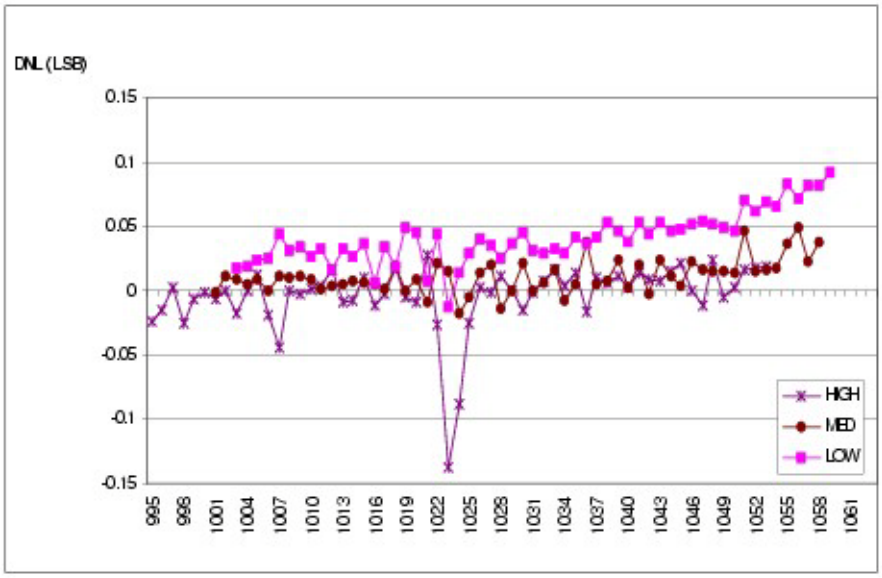


Figure 8. 作为功耗水平的函数的典型 DNL （二阶，CY8C24xxx，5.0V，25° C）



[+] Feedback

Figure 9. 作为功耗水平的函数的典型 DNL （一阶，CY8C29xxx，3.3V，25° C）

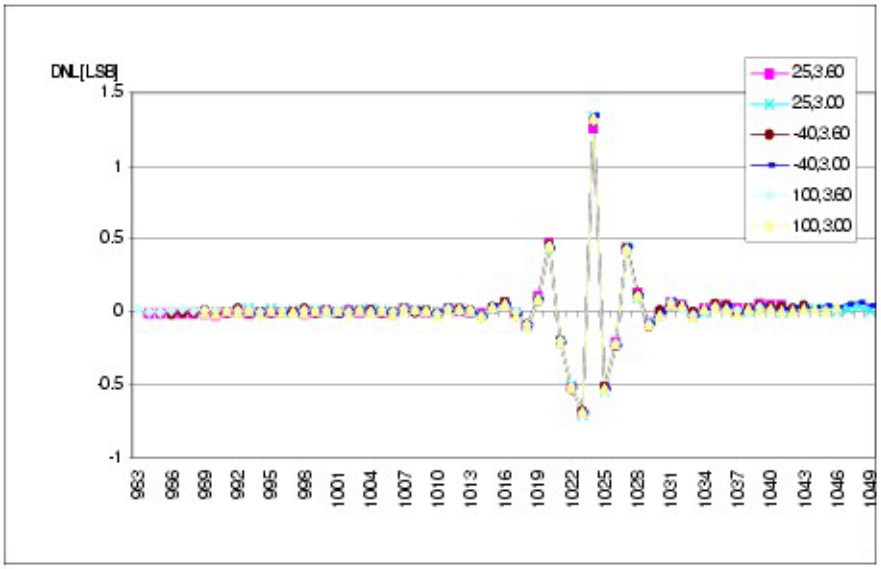
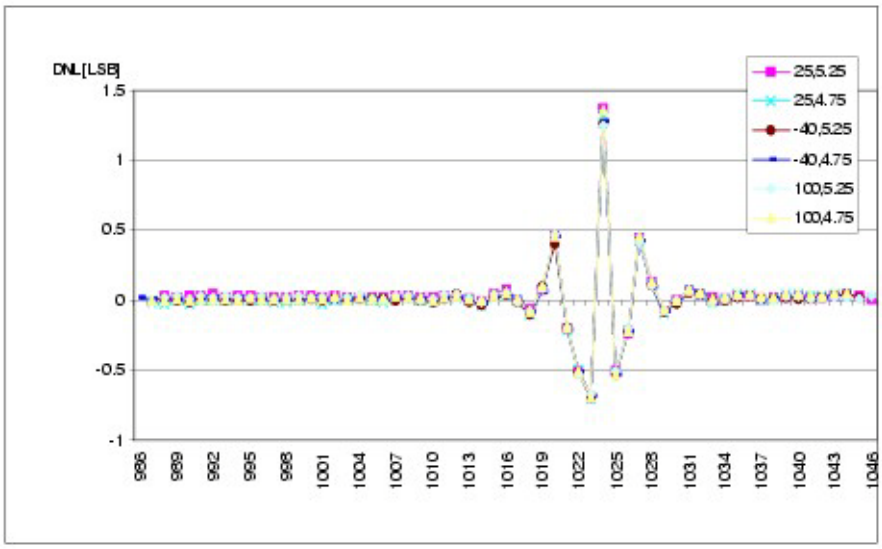


Figure 10. 作为功耗水平的函数的典型 DNL （一阶，CY8C29xxx，5.0V，25° C）



[+] Feedback

Figure 11. 作为功耗水平的函数的典型 DNL （二阶，CY8C29xxx，3.3V，25° C）

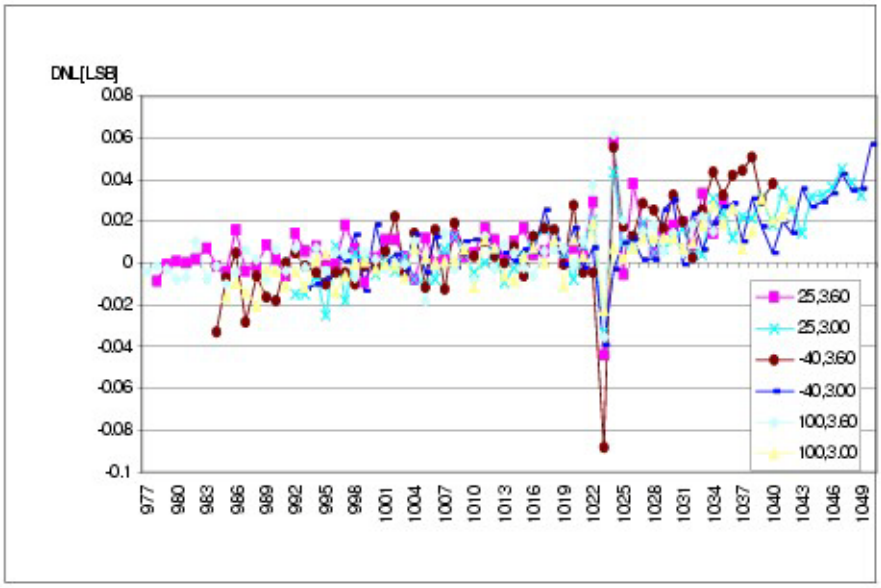
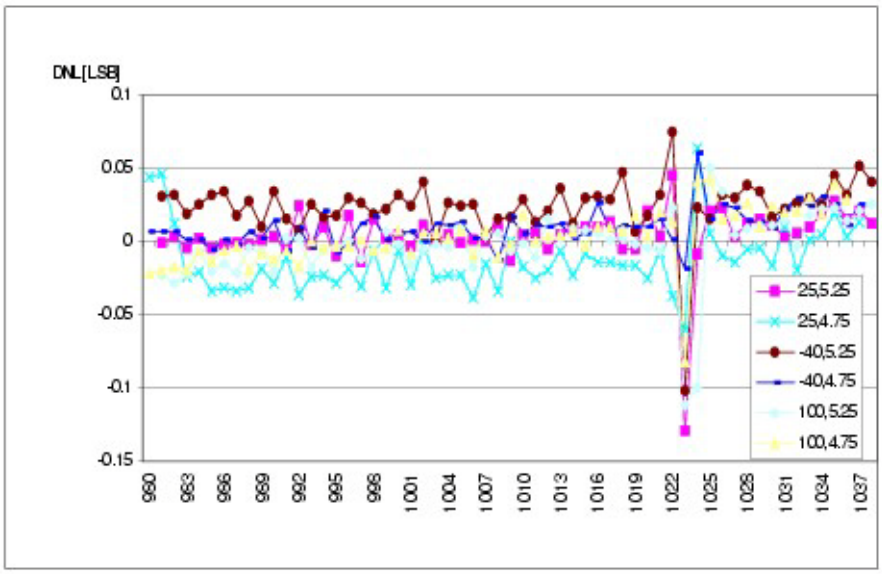


Figure 12. 作为功耗水平的函数的典型 DNL （二阶，5.0V，25° C）



放置

一阶调制器设计需要两个 PSoC 模块，一个为数字模块，另一个为模拟模块。 对于模拟模块的放置，没有任何内部限制；唯一考虑因素是输入和时钟可用性。 不过，数字模块必须能够向硬件抽取滤波器馈送信号。 其放置限制为 CY8C26/25xxx 系列 PSoC 器件中的 DBA02 和 DCA06。 在 CY8C27xxx (A-silicon) 系列中，符合要求的数字模块为 DBB01、DCB02、DBB11 和 DCB12。 在 CY8C29/24/22xxx 和 CY8C27xxx （B-silicon，对于无铅包装，后缀中包含 “X” ）器件系列中，可以使用任意数字模块。 如下所述，两个模块都必须使用同一源时钟。

二阶调制器放置设计与一阶的设计不同，区别之处在于一阶调制器具有另一个开关电容 PSoC 模块。 两个模拟模块必须位于同一列中，以便它们可以共享列比较器总线。 在一阶和二阶调制器中，对数字模块的限制是相同的。

[+] Feedback

虽然模拟和数字模块有很多放置方法，但 DELSIG11 还是使用了 PSoC 器件中仅有的硬件抽取滤波器。 对于给定配置，只能放置一个 DELSIG11 实例。 只要模块不重叠，就可以通过动态重新配置加载多个配置。 虽然两个实例似乎都可以工作，但是只有最近加载的实例的输出是正确的。

参数和资源

一旦放置了 DELSIG11 实例，必须配置四个参数才能正常工作：输入、时钟相位、数据时钟和轮询选择。

数据位置

ADC 生成 11 位数据，它们放置在两个 8-bit 寄存器中。 这些数据或者左对齐，或者右对齐。 右对齐数据的低 8 位存储在最低有效字节中，其高 3 位存储在最高有效字节的低 3 位中。 前 5 位是符号扩展，用于存储数据的 2 的补码格式。 左对齐数据的高 8 位存储在最高有效字节中，其低 3 位存储在最低有效字节的高 3 位中。 低 5 位包含抽取滤波器的其余部分。

输入

此参数确定 ADC 输入的信号源。

时钟相位

选择时钟相位的目的是将一个模拟 PSoC 模块的输出与另一个模块的输入同步。 开关电容模拟 PSoC 模块使用两相时钟（Φ1 和 Φ2）来获取和传输信号。 通常，DELSIG11 的输入是在 Φ1 上采样的。 问题是许多用户模块在 Φ1 中将其输出自动归零，仅在 Φ2 中提供有效输出。 如果此类模块的输出馈送到 DELSIG11 的输入，则 DELSIG11 获取自动归零的输出而不是有效信号。 时钟相位选择允许交换相位，以便在 Φ2 中获取到输入信号。

TMR 时钟

TMR 时钟确定采样率。 此时钟传递到一阶调制器设计的两个 PSoC 模块和二阶设计的所有三个 PSoC 模块。

Note CAUTION: 必须为数字模块和模拟列时钟选择同一时钟，否则此用户模块无法正常工作。

Note 在其他 PSoC 用户模块数据手册中，TMR 时钟称为 “ 数据时钟 ”。

定时器设置为在 TMR 时钟的每 256 个计数处提供一个中断，它使用 4 个中断来处理数据。 因此，采样率的定义如下所示。

Equation 8

$$SampleRate = \frac{DataClock}{1024}$$

示例 1

如果需要 5 ksps 采样率，则 TMR 时钟必须为：

Equation 9

$$DataClock = SampleRate \times 1024 = 5ksps \times 1024 = 5.12MHz$$

CPU 开销也取决于数据时钟。 对 ADC 值进行计算的中断子程序必须为每个输出样品提供 4 个中断。 这 4 个中断中的每个中断在服务子程序中执行一段不同的代码。 第一段和第三段需要 90 个 CPU 周

[+] Feedback

期，第二段需要 154 个 CPU 周期，第四段需要 289 个 CPU 周期。 处理 ADC 数据总共需要 623 个 CPU 周期。 平均而言，在一个采样周期中这 4 个段使用的 CPU 总数为：

Equation 10

$$AverageCPUOverhead = \frac{623 \times Samplerate}{CPUClock}$$

但是，在任何给定的中断周期中，CPU 的使用将因提供该特定中断所需的周期数而异。 每段的开销可以计算为：

Equation 11

$$FirstSectionCPUOverhead = \frac{90 \cdot (4 \cdot SampleRate)}{CPUClock}$$

Equation 12

$$SecondSectionCPUOverhead = \frac{154 \cdot (4 \cdot SampleRate)}{CPUClock}$$

Equation 13

$$ThirdSectionCPUOverhead = \frac{90 \cdot (4 \cdot SampleRate)}{CPUClock}$$

Equation 14

$$FourthSectionCPUOverhead = \frac{289 \cdot (4 \cdot SampleRate)}{CPUClock}$$

开销计算示例

如果采样率为 5 ksps，CPU 时钟为 24 MHz，则 CPU 开销为：

Equation 15

$$TotalCPUOverhead = \frac{623 \times 5ksps}{24MHz} = 13\%$$

[+] Feedback

每段的开销为：

Equation 16

$$\text{First section} = \frac{90 \cdot (4 \cdot 5ksps)}{24MHz} = 7.5\%$$

$$\text{Second section} = \frac{154 \cdot (4 \cdot 5ksps)}{24MHz} = 12.8\%$$

$$\text{Third section} = \frac{90 \cdot (4 \cdot 5ksps)}{24MHz} = 7.5\%$$

$$\text{Fourth section} = \frac{289 \cdot (4 \cdot 5ksps)}{24MHz} = 24.1\%$$

轮询

ADC 的转换时间可能仅有 130 µ?，所以数据的快速检索是非常重要的。 sinc2 滤波器生成的数据是在数字时钟的中断服务子程序 (ISR) 控制下计算的。 有两个选项可用来访问此数据，此数据由可设置为 “ 启用 ” 或 “ 禁用 ” 的轮询属性控制。

当轮询设置为 “ 禁用 ” 时，通过将用来检索数据的代码插入汇编文件中的汇编语言中断子程序 DELSIG11_ADConversion_ISR 来完成数据的检索。 *delsig11INT.asm*. 插入代码的位置被清晰标记。

当轮询设置为 “ 启用 ” 时，将分配两个附加 RAM 变量，一个用于存储数据的副本，另一个用于描述其可用性。 下一节将介绍这些变量和相关轮询函数。

中断生成控制

当在 PSoC Designer 中选中 **启用中断生成控制** 复选框时，会有一个附加参数变为可用。 可以通过依次单击以下菜单获得此参数： **项目 > 设置 > 芯片编辑器**. 当多个外覆层用于由多个用户模块跨外覆层共享的中断时，中断生成控制非常重要：

IntDispatchMode

IntDispatchMode 参数用于指定如何为位于同一模块但在不同外覆层中的多个用户模块共享的中断来处理中断请求。 选择 “ActiveStatus” 会导致固件测试在为共享中断请求提供服务之前哪个外覆层处于活动状态。 每当请求共享中断时，都会发生此测试。 这会增加延迟，还会生成为共享中断请求提供服务的不确定过程，但是不需要任何 RAM。 选择 “OffsetPreCalc” 会导致固件仅当初始加载外覆层时计算共享中断请求的源。 此计算减少了中断延迟，并生成为共享中断请求提供服务的确定过程，但是耗费了一字节 RAM。

[+] Feedback

应用程序编程接口

应用程序编程接口 (API) 子程序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 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。

每次放置用户模块时，都会为其分配一个实例名称。 默认情况下，PSoC Designer 会为指定项目中此用户模块的第一个实例分配 DELSIG11_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 DELSIG11。

全局变量 DELSIG11_bfStatus

说明：

仅当轮询参数的值设置为 “ 启用 ” 时，此变量才可用。 当全局变量 DELSIG11_iResult 包含有效数据时，它设置为非零值。 可以直接访问此变量，也可以通过 API 函数 DELSIG11_ClearFlag、DELSIG11_fIsDataAvailable 和 DELSIG11_GetDataClearFlag. 间接访问此变量。

C 原型：

```
BOOL DELSIG11_bfStatus;
```

全局变量 DELSIG11_iResult

说明：

仅当轮询参数的值设置为 “ 启用 ” 时，此变量才可用。 DELSIG11 转换的每个数据样品放置在此变量中，相关布尔变量 DELSIG11_bfStatus 的值设置为非零值。 可以直接访问此变量的内容，也可以通过 API 函数 DELSIG11_GetData 和 DELSIG11_GetDataClearFlag. 间接访问此变量的内容。另请参见 API 函数 DELSIG11_fIsDataAvailable.

C 原型：

```
int DELSIG11_iResult;
```

DELSIG11_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  DELSIG11_Start (BYTE bPowerSetting)
```

汇编：

```
mov   A, bPowerSetting
lcall  DELSIG11_Start
```

参数：

bPowerSetting： 一个用于指定功耗水平的字节。 在复位和配置后，分配给 DELSIG11 的模拟 PSoC 模块会关闭电源。 下表给出了在 C 语言和汇编语言程序内提供的符号名称及其相关值。

[+] Feedback

符号名称	值
DELSIG11_OFF	0
DELSIG11_LOWPOWER	1
DELSIG11_MEDPOWER	2
DELSIG11_HIGHPOWER	3

返回值：

无

副作用：

A 和 X 寄存器可以由此函数的本次执行或以后执行而被修改。 在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。

DELSIG11_SetPower

说明：

设置开关电容 PSoC 模块的功耗水平。

C 原型：

```
void  DELSIG11_SetPower (BYTE bPowerSetting)
```

汇编：

```
mov   A, bPowerSetting
lcall  DELSIG11_SetPower
```

参数：

bPowerSetting： 与用于起始进入点的 bPowerSetting 参数相同。

返回值：

无

副作用：

A 和 X 寄存器可以由此函数的本次执行或以后执行而被修改。 在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。

DELSIG11_Stop

说明：

将开关电容 PSoC 模块的功耗水平设置为 “ 关闭 ”。

C 原型：

```
void  DELSIG11_Stop (void)
```

汇编：

```
lcall  DELSIG11_Stop
```


[+] Feedback

参数：

无

返回值：

无

副作用：

A 和 X 寄存器可以由此函数的本次执行或以后执行而被修改。 在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。

DELSIG11_StartAD

说明：

启用定时器和积分器。

C 原型：

```
void  DELSIG11_StartAD (void)
```

汇编

```
lcall  DELSIG11_StartAD
```

参数：

无

返回值：

无

副作用：

A 和 X 寄存器可以由此函数的本次执行或以后执行而被修改。 在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。 目前，仅修改 CUR_PP 页指针寄存器。

DELSIG11_StopAD

说明：

禁用定时器并复位积分器。

C 原型：

```
void  DELSIG11_StopAD (void)
```

汇编：

```
lcall  DELSIG11_StopAD
```

参数：

无

返回值：

无

[+] Feedback

副作用：

A 和 X 寄存器可以由此函数的本次执行或以后执行而被修改。 在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。

DELSIG11_fIsDataAvailable

说明：

检查采样数据的可用性。

C 原型：

```
BYTE  DELSIG11_fIsDataAvailable(void)
```

汇编：

```
lcall  DELSIG11_fIsDataAvailable
cmp   A, 0
jz    .DataNotAvailable
```

参数：

无

返回值：

如果数据已转换且可以读取，则返回非零值。

副作用：

A 和 X 寄存器可以由此函数的本次执行或以后执行而被修改。 在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。 目前，仅修改 CUR_PP 页指针寄存器。

DELSIG11_iGetData

说明：

返回转换的数据。应当调用 DELSIG11_fIsDataAvailable() 以验证数据采样是否已就绪。 如果对此函数的调用正好在积分周期结束时完成，则返回的数据有可能损坏。 因此，极力建议以高于采样率的频率进行数据检索，或者如果不能保证在调用此函数之前关闭中断，则应进行数据检索。

C 原型：

```
INT  DELSIG11_iGetData(void)
```

汇编：

```
lcall  DELSIG11_iGetData          ; LSB will be in A, MSB in X upon return
```

参数：

无

返回值：

以 8-bit 2 的补码格式返回转换的数据采样。

[+] Feedback

副作用：

A 和 X 寄存器可以由此函数的本次执行或以后执行而被修改。 在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。 目前，仅修改 CUR_PP 页指针寄存器。

DELSIG11_ClearFlag

说明：

复位数据可用标志。

C 原型：

```
void  DELSIG11_ClearFlag(void)
```

汇编：

```
lcall  DELSIG11_ClearFlag
```

参数：

无

返回值：

无

副作用：

全局变量 DELSIG11_bfStatus 设置为零。可以通过此函数的此实现或将来实现修改 A 和 X 寄存器。在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。 目前，仅修改 CUR_PP 页指针寄存器。

DELSIG11_iGetDataClearFlag

说明：

返回转换的数据并复位数据可用标志。应当调用 DELSIG11_fIsDataAvailable() 以验证数据采样是否已就绪。 如果对此函数的调用正好在积分周期结束时完成，则返回的数据有可能损坏。 因此，极力建议以高于采样率的频率进行数据检索，或者如果不能保证在调用此函数之前关闭中断，则应进行数据检索。

C 原型：

```
INT  DELSIG11_iGetDataClearFlag(void)
```

汇编：

```
lcall  DELSIG11_iGetDataClearFlag   ; LSB will be in A, MSB in X upon return
```

参数：

无

返回值：

以 8-bit 2 的补码格式返回转换的数据采样。

副作用：

全局变量 DELSIG11_bfStatus 设置为零。 A 和 X 寄存器可以由此函数的本次执行或以后执行而被修改。 在大内存模式下 (CY8C29xxx)，所有 RAM 页指针寄存器也会出现这种情况。 如果需要，由调用函数负责将调用前后的数值保存在 fastcall16 函数内。 目前，仅修改 CUR_PP 页指针寄存器。

[+] Feedback

固件源代码示例

示例 1：启用轮询，直接访问

在此示例中，轮询用于确定样品何时可用。 通过直接访问全局变量可获得最大速度。 调用伪子程序表示将进一步处理从 DELSIG11 ADC 获取的样品。

下面是汇编语言示例。

```
include "m8c.inc"       ; part specific constants and macros
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:

   M8C_EnableGInt                ; enable global interrupts
   mov   A,DELSIG11_HIGHPOWER    ; Establish power setting...
   call  DELSIG11_Start          ; and initialize
   call  DELSIG11_StartAD        ; Commence sampling process
mainloop:
   cmp   [DELSIG11_bfStatus], 0
   jz    mainloop                ; spin lock until(data is Available)
   mov   [DELSIG11_bfStatus], 0  ; reset the data available flag
   mov   X, [DELSIG11_iResult+0] ; grab valid data and pass using fast-
   mov   A, [DELSIG11_iResult+1] ;   call convention (LSB in A, MSB in X)
   call  ProcessSample           ; pass the sample in A to the dummy fcn
   jmp   mainloop


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

下面是等效的 C 语言代码：

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

void ProcessSample( int iSample )
{
    ; // (Do something useful with the data)
}

void main(void)
{
   extern BYTE DELSIG11_bfStatus;
   extern int  DELSIG11_iResult;
   M8C_EnableGInt;
   DELSIG11_Start( DELSIG11_HIGHPOWER );
   DELSIG11_StartAD();
   while (1) {
      if ( DELSIG11_bfStatus ) {
         DELSIG11_bfStatus = 0;
         ProcessSample( DELSIG11_iResult );
      }
```

```
    }
}
```

示例 2：启用轮询，间接访问

此示例重复上一示例情形，但是使用 API 函数而不是直接引用全局变量。

下面是汇编语言示例。

```
include "DELSIG11.inc"
include "m8c.inc"

main:
   M8C_EnableGInt                  ; enable global interrupts
   mov   A,DELSIG11_HIGHPOWER      ; Establish power setting...
   call  DELSIG11_Start            ; and initialize
   call  DELSIG11_StartAD          ; Commence sampling process
mainloop:
   call  DELSIG11_fIsDataAvailable ; Retrieve the status byte
   cmp   A, 0
   jz    mainloop                  ; spin lock until(data is Available)
   call  DELSIG11_GetDataClearFlag ; fastcall convention puts data in X, A
   call  ProcessSample             ; pass the sample to the dummy fcn
   jmp   mainloop


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

同样。下面是等效的 C 语言代码：

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

void ProcessSample( int iSample )
{
    ; // (Do something useful with the data)
}

void main(void)
{
   M8C_EnableGInt;
   DELSIG11_Start( DELSIG11_HIGHPOWER );
   DELSIG11_StartAD();
   while (1) {
      if ( DELSIG11_fIsDataAvailable() ) {
         ProcessSample( DELSIG11_iGetDataClearFlag() );
      }
   }
}
```


[+] Feedback

配置寄存器

Table 7. “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
CR0	1	0	0	1	0	0	0	0
CR1	InputSource			0	0	0	0	0
CR2	0	1	AZ	0	0	0	0	0
CR3	1	1	1	FSW0	0	0	0	0

ADC 是开关电容 PSoC 模块。 它配置为生成模拟调制器。 要生成调制器，需要将该模块配置为带有参考反馈的积分器，该积分器将输入值转换为数字脉冲流。 输入复用器确定将对哪些信号执行数字化。

InputSource 字段选择由转换器执行数字化的输入信号。 此参数是在设备编辑器中设置的。

TMR 中断处理程序和各种 API 使用 AZ 和 FSW0 来复位积分器。

Table 8. TMR 数字 PSoC 模块使用的寄存器

寄存器	7	6	5	4	3	2	1	0
功能	0	0	1	0	0	0	0	0
输入	0	0	0	1	时钟			
输出	0	0	0	0	0	0	0	0
DR0	定时器倒计数值 （API 从不访问）							
DR1	1	1	1	1	1	1	1	1
DR2	不使用							
CR0	0	0	1	0	0	0	0	启用

TMR 是所配置的定时器具有 256 个计数周期的数字 PSoC 模块。 在中断时，会读取抽取滤波器并计算 ADC 值。

时钟从 16 个源之一选择输入时钟。 此参数是在设备编辑器中设置的。 请注意，还必须使用选择的源来控制 ADC 模块驻留的列的模拟时钟。

设置为 “ 启用 ” 后，TMR 便可以工作了。 它由 DELSIG11 API 修改和控制。

Table 9. 抽取控制寄存器

位	7	6	5	4	3	2	1	0
DEC_CR	0	0	1	0	0	0	DCol	DCLKSEL
DEC_DH	抽取滤波器的高字节输出							
DEC_DL	抽取滤波器的低字节输出							

抽取滤波器是用于实现 ΔΣ ADC 所需的 Sinc2 滤波器的专用硬件。 它由一个控制寄存器和两个数据输出寄存器组成。 当 DR0 中的值倒数到终端计数时，将调用中断以降低软件计数器和 CNT 从 DR1 重新加载的较高值。 数据通过 DR2 输出。

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。 这两个参数都是在设备编辑器中设置的。

[+] Feedback

版本历史记录

版本	创作者	说明
3.2	DHA	增加了 DRC 以检查是否出现下列情况： 1. 数字和模拟资源的源时钟不同。 2. ADC 时钟大于 CPU 时钟。

Note PSoC Designer 5.1 在所有的用户模块数据手册中提供版本历史记录。 本数据表详细介绍了当前和先前用户模块版本之间的区别。

Copyright © 2002-2011 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.