易于使用的 ADC 数据手册 EzADC V 1.00

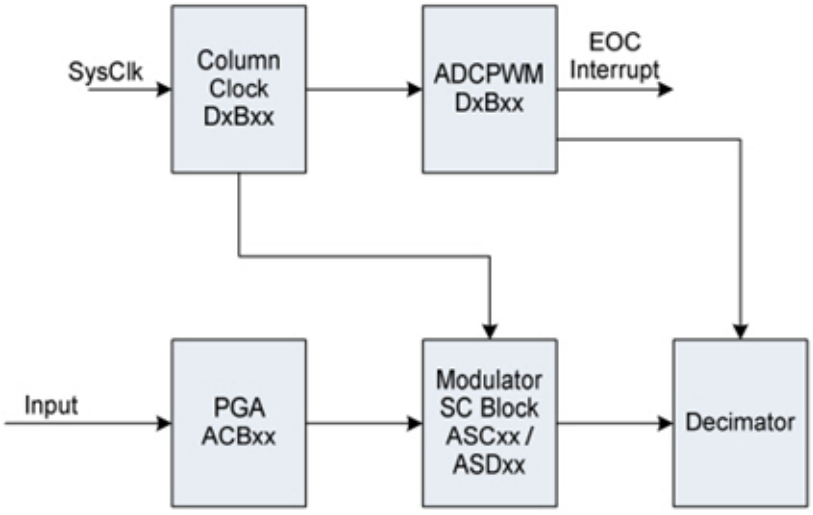
Copyright © 2012-2014 Cypress Semiconductor Corporation. All Rights Reserved.

MUM	PSoC® 模块				API 存储器 （字节）				引脚
	数字	模拟 CT	模拟 SC		闪存		RAM		
调制器 （一阶或二阶）		一阶和二阶	一阶	二阶	一阶	二阶	一阶	二阶	
支持的器件包括：CY8C24x23A、 CY8C27x43、 CY8C24x94、 CY8C29x66、 CY8C28x23、 CY8C28x33、 CY8C28x43、 CY8C28x45、 CY8C28x52									
配置	2	1	1	2	365	413	11	11	1

特性和概述

- 增量型 ADC 支持一阶和二阶调制器
- 可选采样率：0.007 ~ 15.625 ksps
- 可选分辨率：6 ~ 14 位
- 可选增益：1x ~ 48x
- 可选参考
- 自动计算时钟频率
- 有符号和无符号的输出数据格式
- 允许 AGND 输出连接至 AnalogBus
- 补偿偏移误差
- 可配置补偿偏移误差频率

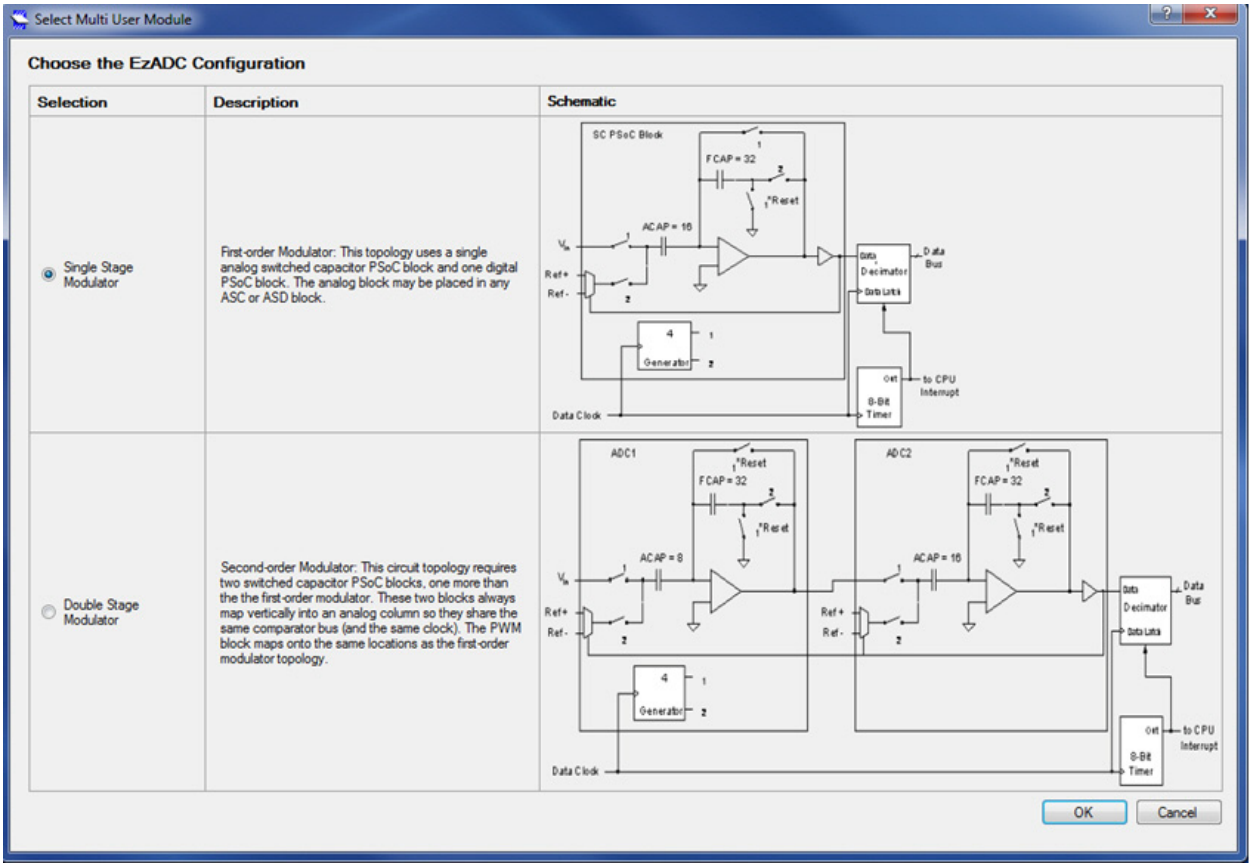
图 1. EzADC 框图



功能说明

EzADC 用户模块具有两种配置方式：一阶调制器和二阶调制器。

图 2. EzADC MUM 向导



一阶调制器使用单模拟开关电容 PSoC 模块，如图 2 所示。模拟模块可放置在任何模拟开关电容 （ASC）或一个 ASC D 类型 （ASD）模块中。

二阶调制器则需要两个开关电容 PSoC 模块。因为这两个模块始终垂直地映射到模拟列，他们共享了同一个比较器 （以及同一个时钟）。 EzADC 用户模块包含一个可编程增益放大器 （PGA）、一个时钟分频器（计数器 8）、一个抽取滤波器 （积分器）、一个 ADC 调制器 （根据选定的阶次，它可以使用一个或两个 SC 模块）、一个脉冲宽度调制器 （PWM8）以及 AnalogClock_0_Select MUX.

系统时钟 （SysClk）除为时钟分频器提供时钟源外，还为 PWM 和 SC 模块提供时钟源。时钟分频器的输出被路由给模拟列和 PWM。用户模块使用两个连续数字模块。

ADC 由调制器 （模拟 SC 模块）、抽取滤波器和 PWM 构成。根据选定的分辨率，使用 PWM 来设置积分时间。转换结束时， PWM 会锁存抽取滤波器输出，并生成中断，表示转换结束。

ADC SC 调制器模块的正向输入连接到 CT 模块的输出端， CT 模块与 SC 模块位于同一列上。将 CT 模块配置为 PGA。可以在用户模块参数窗口中对 CT 模块的参考和输入进行配置。在 Interconnect View — Input （互联试图 — 输入）中， EzADC 用户模块仅有一个可编程输入。可能的 ADC 输入值由 PMux 定义，而该 PMux 则取决于所使用的 ACB 模块。

所有模拟模块始终位于同一个模拟列内。 AnalogColumn_Clock MUX 始终被连接至 AnalogClock_0_Select MUX。带有一阶调制器的 EzADC 用户模块可以放置在任何列的顶部和中间模块内。带有二阶调制器的用户模块使用的是模拟列中的三个模拟模块。

直流和交流电气特性

表 1. EzADC 电气特性

参数	最小值	典型值	最大值	单位	条件和注意
V_{IN}	–	–	V_{SS} 到 V_{DD}	V	
分辨率	–	–	–	位	
采样率	–	–	–	ksps	
G_{48}	–	3.0	–	%	缓冲器增益 = 48；偏移比较器使能
G_{24}	–	2.2	–	%	缓冲器增益 = 24；偏移比较器使能
G_{16}	–	1.5	–	%	缓冲器增益 = 16；偏移比较器使能
G_4	–	0.7	–	%	缓冲器增益 = 4；偏移比较器使能
G_1	–	0.5	–	%	缓冲器增益 = 1；偏移比较器使能
V_{OS}	–	4.5	–	mV	缓冲器增益 = 1；偏移比较器禁用
$V_{OS\ COMP}$	–	0.5	–	mV	缓冲器增益 = 1；偏移比较器使能
$I_{leakage}$	–	1	–	nA	
C_{IN}	–	3	–	pF	
PSRR	–	73	–	dB	
I_{oper}	–	272 1380 5452	–	µA	低功耗中等功耗高功耗
DNL	–	0.1	–	LSB	列时钟为 2 MHz
INL	–	0.5	–	LSB	列时钟为 2 MHz
SNR	–	46	–	dB	

放置

带有一阶调制器的 EzADC 需要一个 CT 模块、一个 SC 模块和两个数字模块。可以将模拟模块放置在任何占用顶部和中间行的模拟列内。两个数字模块在任意数字行上占用了连续的模块。而带有二阶调制器的 EzADC 则需要一个 CT 模块和两个 SC 模块，因此它占用了整个模拟列。

参数和资源

ADC 分辨率

该参数将设置 ADC 的分辨率。范围为 6 至 14 位。

ADC 采样率

该参数将设置 ADC 的采样率。

分辨率	采样率选择 （ksps）
6	3.906、 1.953、 0.976
7	2.604、 1.302、 0.651
8	1.562、 0.781、 0.390
9	0.868、 0.434、 0.217
10	0.459、 0.229、 0.114
11	0.236、 0.118、 0.059
12	0.120、 0.060、 0.030
13	0.060、 0.030、 0.015
14	0.030、 0.015、 0.007

注意： 通过使用以下公式，可根据分别为 2 MHz、 1 MHz 和 500 KHz 的数据时钟频率，对上述所有采样率进行计算：

$$SampleRate = \frac{DataClock}{256 \cdot (2^{Bits-6} + 1)}$$

缓冲器增益

该参数设置了 PGA 的增益。可用的增益选项分别为 48.000、 24.000、 16.000、 8.000、 5.333、 4.00、 3.200、 2.667、 2.286、 2.000、 1.777、 1.600、 1.455、 1.333、 1.231、 1.143、 1.062 和 1.000。

参考

通过该参数为 PGA 设置参考输出。 VSS 和 AGND 是参考选项

数据格式

通过该参数设置数据格式。可以将数据设置为有符号或无符号格式。

AGND 输出

使用该参数可允许或禁止将 AGND 输出连接至模拟总线。使用该选项可生成内部 AGND，以便将双极信号连接至 ADC。

补偿偏移

使用该参数可启用或禁用相关二次采样。关于相关二次采样的更多信息，请参考 AN2226。为了补偿偏移，需将 PGA 参考输出设置为 AGND、将 PGA 输入连接至 AGND，并执行单一转换。将 ADC 的输出存储在 RAM 变量中。在其他测量中，要从 ADC 结果中减去该偏移值。

补偿偏移频率

通过该参数可设置 ADC 采样的数量，而采样后偏移会被更正。二次采样后可通过将该参数设置为 1 来获取最佳结果，但该操作会降低一半 ADC 的吞吐量。

应用编程接口 （API）

所提供的应用编程接口 （API）子程序作为用户模块的一部分，设计人员通过它可以采用更高级的方式处理模块。本节指定了每个函数的接口，以及 “include” 文件所提供的相关常量。

注意：

在这里，同所有用户模块 API 中的一样， A 和 X 寄存器的值可以通过调用 API 函数进行更改。如果在调用后需要 A 和 X 的值，则调用函数需要保留调用前的 A 和 X 的值。选择该 “ 寄存器易失 ” 策略是为了提高效率，自 PSoC Designer 1.0 版起便强制使用该策略。 C 编译器自动遵守该要求。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 函数可以保持 A 和 X 不变，但是无法保证它们将来也会如此。

EzADC_Start

说明：

用于初始化并启动 EzADC 用户模块源。

C 原型：

```
void  EzADC_Start (BYTE bPowerSetting)
```

汇编：

```
mov   A, bPowerSetting
lcall  EzADC_Start
```

参数：

bPowerSetting：设置 ADC 和 PGA 运行功率。

符号名称	数值
EzADC_LOWPOWER	1
EzADC_MEDPOWER	2
EzADC_HIGHPOWER	3

返回值：

无

EzADC_Stop

说明：

停止 EzADC 用户模块源。

C 原型：

```
void  EzADC_Stop (void)
```

汇编：

```
lcall  EzADC_Stop
```

参数：

无

返回值：

无

EzADC_SetBufferGain

说明：

设置 PGA 的增益。

符号名称	数值
EzADC_G48_00	0Ch
EzADC_G24_00	1Ch
EzADC_G16_00	08h
EzADC_G8_00	18h
EzADC_G5_33	28h
EzADC_G4_00	38h
EzADC_G3_20	48h
EzADC_G2_67	58h
EzADC_G2_27	68h
EzADC_G2_00	78h
EzADC_G1_78	88h
EzADC_G1_60	98h
EzADC_G1_46	A8h
EzADC_G1_33	B8h
EzADC_G1_23	C8h
EzADC_G1_14	D8h
EzADC_G1_06	E8h
EzADC_G1_00	F8h

C 原型：

```
void EzADC_SetBufferGain(BYTE bGainSetting)
```

汇编

```
mov A, bGainSetting
lcall EzADC_SetBufferGain
```

参数

无

返回值

无

EzADC_GetSamples

说明：

启动 ADC 转换。

C 原型：

```
void EzADC_GetSamples (BYTE bNumSamples)
```

汇编：

```
mov A, bNumSamples
lcall EzADC_GetSamples
```

参数：

无

返回值：

无

EzADC_fIsDataAvailable

说明：

该函数用于检测 EzADC 转换是否完成。

C 原型：

```
BYTE EzADC_fIsDataAvailable(void)
```

汇编：

```
lcall EzADC_fIsDataAvailable
```

参数：

无

返回值：

如果未完成转换，则返回 0；如果转换已完成，则返回 1。

EzADC_iGetDataClearFlag

说明：

读取 ADC 结果，并清除 ADC 就绪标志。

C 原型：

```
INT   EzADC_iGetDataClearFlag(void)
```

汇编：

```
lcall  EzADC_iGetDataClearFlag
```

参数：

无

返回值：

以整数格式返回 ADC 结果。

EzADC_SetReference

说明：

为 PGA 设置参考。

C 原型：

```
void EzADC_SetReference(BYTE bReferenceSetting)
```

汇编：

```
mov A, bReferenceSetting
lcall EzADC_SetReference
```

参数：

bReference：PGA 的参考

符号名称	数值
EzADC_REF_VSS	0x10
EzADC_REF_AGND	0x01

返回值

无

固件源代码示例

下面介绍的是该用户模块的 C 语言代码示例。

```
//-------------------------------------------------------------------------
// C main line
//-------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
int ADCResult;
void main(void)
{
    M8C_EnableGInt;                             // Enable Global Interrupts
    EzADC_Start(EzADC_HIGHPOWER);               // Apply power to ADC
    EzADC_SetBufferGain(EzADC_G2_00);           // Set Gain
    EzADC_GetSamples(0);                        // Have ADC run continuously
    for(;;)
    {
        while (!(EzADC_fIsDataAvailable()));    // Loop until value ready
        ADCResult = EzADC_iGetDataClearFlag();  // Reset ready flag and get data
        // Add user code here to use or display result
    }
}
```

下面内容是写入汇编语言的等效代码：

```
//-------------------------------------------------------------------------
//  Assembly main line
//-------------------------------------------------------------------------
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants and macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

area bss(RAM)
ADCResult    : blk 2

area text(ROM,REL)
_main:

    M8C_EnableGInt                  ; Enable Global Interrupts
    mov A, EzADC_HIGHPOWER
    lcall EzADC_Start               ; Apply power to ADC

    mov A, EzADC_G2_00
    lcall EzADC_SetBufferGain       ; Set Gain

    mov A, 0
    lcall EzADC_GetSamples       ; Have ADC run continuously
loop1:
wait:
    lcall EzADC_fIsDataAvailable ; Loop until value ready
    jz wait
    call EzADC_iGetDataClearFlag ; Reset ready flag and get data
```

```
    mov [ADCResult], X
    mov [ADCResult + 1], A
    ; Add user code here to use or display result

jmp loop1
```

版本历史记录

版本	创作者	说明
1.0	KUK	第一版本

注意： PSoC Designer 5.1 在所有用户模块基本介绍中都引入了 “ 版本历史 ”。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 © 2012-2014 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。