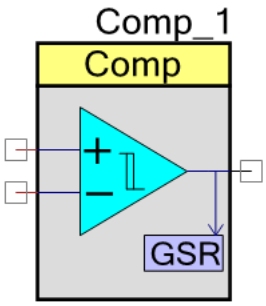
PSoC 4 电压比较器（Comp）

1.10



特性

- 低输入偏移
- 用户控制的偏移校准
- 多种速度模式
- PSoC 4200 BLE 可以在深度睡眠模式下运行
- 输出可路由至数字逻辑模块或引脚
- 可选择输出极性
- 多中断模式

概述

PSoC 4 电压比较器（或比较器）组件提供了硬件解决方案，用以比较两路模拟输入电压。输出可通过软件采样，或者路由至数字组件。组件提供了三个速度级别，这样您能够优化速度或功耗。可以将内部参考电压或外部电压连接到任何输入端。

在全温度和全电压范围内，输入偏移电压小于 1 mV。可以选择无迟滞或带 10 mV 的迟滞。

对于 PSoC 4200 BLE 器件，比较器可以在深度睡眠模式下运行。

何时使用比较器

与使用 ADC 相比，比较器可以提供快速比较两个电压的功能。虽然 ADC 可以与软件配合使用进行比较多个电压电平，但是比较器更适用于需要快速响应或很少软件干预的应用场合。应用示例包括：CapSense®、电源或从模拟电平到数字信号的简单转换。

通常的配置是将电压 DAC 连接到负输入终端来创建可调整比较器。

输入/输出连接

本节介绍了比较器的各种输入和输出连接。

正端输入 — 模拟输入

此输入通常连接到需要比较的电压。此输入可从 GPIO 或内部来源路由选择。

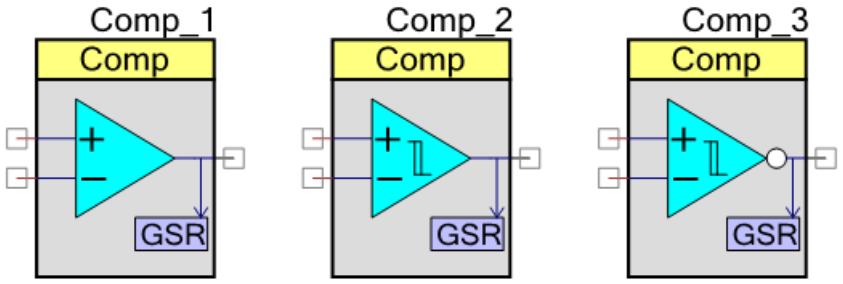
负端输入 – 模拟输入

此输入通常连接到参考电压。此输入可从 GPIO 或内部来源路由选择。

比较器输出 — 输出

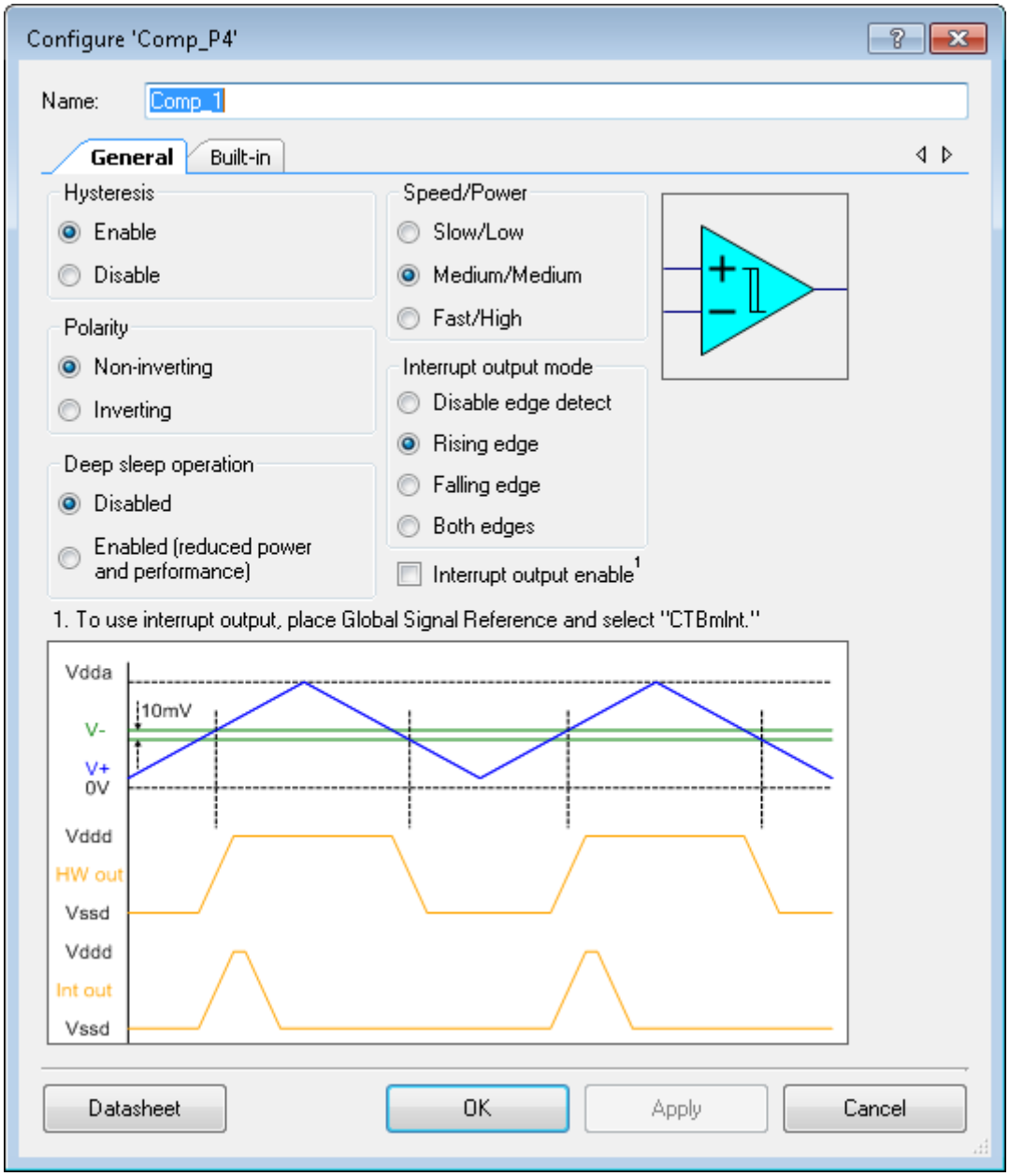
这是数字比较输出。对于非反转配置，当正端输入电压大于负端输入电压时，此输出变为高电平。如果极性设置为反转，则当负端输入电压大于正端输入电压时，输出变为高电平。反转配置可通过使用 UDB 块中的反相器实现，因此需要具有 UDB 资源的设备。输出可路由至其他组件的数字输入如中断、定时器，等等。

此组件标注的符号表示选择输出电压迟滞或反转。



组件参数

将比较器拖入到设计中，然后双击它，以打开 Configure 对话框。

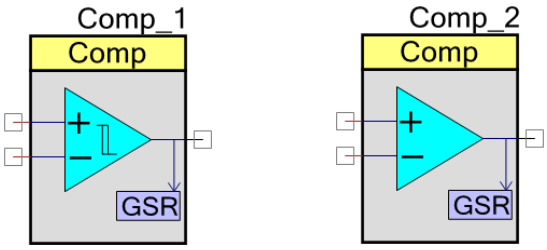


组件包含下列参数。

Hysteresis（迟滞）

通过此参数，可以使比较器具备大约 10 mV 的迟滞。当两个输入电压几乎相等时，有助于确保缓慢变化的电压或稍有噪声的电压，不会导致输出振荡。

使能迟滞 禁用迟滞



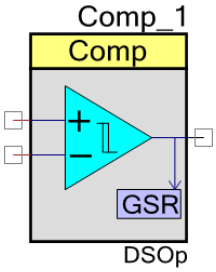
Polarity（极性）

可以通过该参数反转输出。这对于需要来自比较器反转信号的外设非常有用。通过软件 API 返回的采样信号状态和通过电源管理器显示的输出（请参考*系统参考指南*中的 Alt 活动和睡眠一节）不受该参数的影响。

注意：比较器的反转逻辑可通过使用 UDB 资源实现。使能深度睡眠模式时，该选项无效。

Deep sleep operation（深度睡眠操作）

该参数仅适用于 PSoC 4200 BLE 器件。它允许组件在深度睡眠模式下运行。如果使能了该选项，那么“DSOp”标签会显示在符号下方。



注意：当使能深度睡眠模式并且组件供电电压小于 2.7 V 时，**Speed/Power**（速度/功耗）参数只能被设置为 Slow/Low（慢速/低）。High（高）和 Medium（中）的设置情况会引起输出振荡。

Speed/Power（速度/功耗 ）

此参数为您提供了优化速度和功耗的选项。**Speed/Power**（速度/功耗）包括以下选项：Fast Speed/High Power（快速/高功耗）、Medium Speed/Medium Power（中速/中等功耗）和 Slow Speed/Low Power（慢速/低功耗）。

Interrupt output mode（中断输出模式）

该参数定义了导致在比较器中断输出上生成脉冲的事件。中断模式包括以下选项：Disable edge detect（禁用边沿检测）、Rising edge（上升沿）、Falling edge（下降沿）或 Both edges（双边沿）。

Interrupt output enable（中断输出使能）

通过使用该参数，您可以为 PSoC 4200 BLE 配置中断屏蔽。

注意：只有使能了 **Interrupt output enable** （中断输出使能）选项，才能生成比较器中断输出上的脉冲。另外，Comp_SetInterruptMask() API 或 Comp_EnableInterruptOutput() API 函数也可以用于此目的。

应用编程接口（API）

通过应用编程接口（API），您可以使用软件对组件进行配置。该表列出了每个函数的接口，并进行了说明。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“Comp_1”分配给已有设计中的第一个组件实例。可将该值更改为符合标识符语法规则的任意唯一值。该实例名称成为每个全局函数名称、变量和常量符号的前缀。为了提高可读性，下表中使用了实例名称“Comp”。

通用函数

函数	说明
Comp_Start()	对组件执行所有必要的初始化操作，并给模块上电。
Comp_Stop()	关闭比较器模块。
Comp_Init()	根据自定义程序“Configure”对话框的设置内容，初始化或恢复组件。
Comp_Enable()	激活硬件，并开始组件操作。
Comp_GetCompare()	返回比较结果。
Comp_SetSpeed()	将功耗速度等级设置为以下某种设置（共三种）：SLOW_SPEED（低速率）、MED_SPEED（中速率）或HIGH_SPEED（高速率）。
Comp_ZeroCal()	执行输入偏移的自定义校准，使在特定条件下的误差最少。
Comp_LoadTrim()	该函数会将一个数值写入到比较器调整寄存器内。

中断输出函数

函数	说明
Comp_SetInterruptMode()	设置中断沿检测模式。
Comp_EnableInterruptOutput()	使能当前比较器实例的输出。
Comp_DisableInterruptOutput()	禁用当前比较器实例的中断输出。
Comp_GetInterruptOutputStatus()	获取当前实例的中断状态
Comp_ClearInterruptOutput()	清除待处理的中断
Comp_SetInterruptOutput()	设置中断输出。
Comp_GetInterruptSource()	返回所有待处理比较器中断的状态
Comp_ClearInterrupt()	使用掩码清除所有比较器中断的中断
Comp_SetInterrupt()	使用掩码为所有比较器设置软件中断请求。
Comp_SetInterruptMask()	为所有比较器设置中断掩码
Comp_GetInterruptMask()	返回中断掩码。
Comp_GetInterruptSourceMasked()	返回由中断掩码屏蔽的中断请求寄存器。

低功耗函数

函数	说明
Comp_Sleep()	这是让组件准备进入睡眠状态的首选API。
Comp_Wakeup()	该API函数用于将组件恢复到调用Comp_Sleep()前的状态。

void Comp_Start(void)

说明： 对组件执行所有必要的初始化操作，并给模块上电。第一次执行子程序时，需要设置速度/功耗等级和迟滞。在调用Stop()后，重启比较器会保留当前组件的参数设置。

参数： 无

返回值： 无

其他影响： 无

void Comp_Stop(void)

说明： 关闭比较器模块。

参数： 无

返回值： 无

其他影响: 不会影响比较器模式或功耗设置

void Comp_Init(void)

说明： 根据自定义程序“Configure”对话框的设置情况，进行初始化或恢复组件。无需调用Init()，因为Start() API会调用该函数，这是开始组件操作的优选方法。

参数： 无

返回值： 无

其他影响： 根据自定义程序“Configure”对话框中的内容设置所有寄存器。

void Comp_Enable(void)

说明： 激活硬件并开始执行组件操作。无需调用Enable()，因为Start() API会调用该函数，这是开始组件操作的优选方法。

参数： 无

返回值： 无

其他影响： 无

uint32 Comp_GetCompare(void)

说明： 当连接到正端输入的电压大于负端输入电压时，该函数返回非零值。该值不受“极性”参数的影响。该值始终反映了非反转状态配置。

该函数读取直接（unflopped）比较器输出，该输出可能是亚稳态的（因为它会导致错误数据）。

参数： 无

返回值： 比较器输出状态。当正输入电压大于负输入电压时会返回非零值；否则返回值为零。

其他影响： 无

void Comp_SetSpeed(uint32 speed)

说明： 将速度设置为以下三种设置中的一种：慢速，中速或快速。

参数： (uint32) speed: Comp_SLOW_SPEED, Comp_MED_SPEED, Comp_HIGH_SPEED

返回值： 无

其他影响： 无

uint32 Comp_ZeroCal(void)

说明： 执行输入偏移的自定义校准，以使特定条件下的误差最少：比较器参考电压、供电电压和工作温度。执行偏移校准时，一个在该比较器的使用范围中的参考电压必须连接到比较器负输入端。可以通过外部设备或在正端输入上使用内部模拟复用器来完成，即在正常操作时正端输入信号和校准时的负端输入信号之间选择。

参数： 无

返回值： (uint32)偏移校准完成后比较器将调整寄存器中的值。该值的格式与LoadTrim() API子程序的输入参数相同。

其他影响: 在校准过程期间，比较器输出可能无规律。在校准期间，可忽略比较器输出。

void Comp_LoadTrim(uint32 trimVal)

说明： 该函数将某一值写入到比较器偏移调整寄存器。

参数： uint32 trimVal：存储在比较器偏移调整寄存器中的值。该值的格式与ZeroCal()子程序返回值的格式相同。

返回值： 无

其他影响： 无

void Comp_SetInterruptMode(uint32 mode)

说明： 设置中断沿检测模式。

参数： mode：枚举沿检测模式值。 请参考下面的表格。

名称	说明
Comp_INTR_DISABLE	禁用边沿检测
Comp_INTR_RISING	上升沿检测
Comp_INTR_FALLING	下降沿检测
Comp_INTR_BOTH	检测双边沿

返回值： 无

其他影响： 无

void Comp_EnableInterruptOutput (void)

说明： 使能中断输出，使它与全局信号CTBmInt进行OR运算，既为它与其他比较器中断输出进行OR运算。CTBmInt来自于全局信号组件。

参数： 无

返回值： 无

其他影响： 无

void Comp_DisableInterruptOutput (void)

说明： 禁用中断输出，使它不能与全局信号CTBmInt进行OR运算，既为它不能与其他比较器中断输出进行OR运算。

参数： 无

返回值： 无

其他影响： 无

uint32 Comp_GetInterruptOutputStatus (void)

说明： 将返回比较器中断输出的状态。

参数： 无

返回值： 0 = 中断无效，1 = 中断有效

其他影响: 无

void Comp_ClearInterruptOutput (void)

说明： 清除中断请求。该函数与来自全局参考信号的组合中断信号一起使用。

参数： 无

返回值： 无

其他影响： 无

void Comp_SetInterruptOutput (void)

说明： 设置软件中断请求。该函数与来自全局参考信号的组合中断信号一起使用。

参数： 无

返回值： 无

其他影响： 无

uint32 Comp_GetInterruptSource(void)

说明： 获得中断请求。该函数与来自全局参考信号的组合中断信号一起使用。该函数来自任意一个组件实例，可用于确定CTBm模块中所有比较器中断的中断源。

参数： 无

返回值： 中断源。每个组件实例均有一个掩码值：Comp_INTR。

其他影响: 无

void Comp_ClearInterrupt(uint32 interruptMask)

说明： 清除中断请求。该函数与来自全局参考信号的组合中断信号一起使用。该函数可用于清除CTBm模块中任意一个或两个中断。

参数： interruptMask：需要清除的中断的掩码。每个组件实例均有一个掩码值：Comp_INTR。

返回值： 无

其他影响： 无

void Comp_SetInterrupt(uint32 interruptMask)

说明： 设置软件中断请求。该函数与来自全局参考信号的组合中断信号一起使用。该函数可用于触发CTBm模块中的任意一个或两个软件中断。

参数： interruptMask：需要设置的中断的掩码。每个组件实例均有一个掩码值：Comp_INTR。

返回值： 无

其他影响： 无

void Comp_SetInterruptMask (uint32 interruptMask)

说明： 为CTBm模块中的所有比较器配置中断掩码。

参数： interruptMask：激活中断源的位掩码。每个组件实例均有一个掩码值：Comp_INTR_MASK。

返回值： 无

其他影响： 无

uint32 Comp_GetInterruptMask (void)

说明： 为CTBm模块中的所有比较器返回中断掩码。

参数： 无

返回值： 启用中断源的掩码。每个组件实例均有一个掩码值：Comp_INTR_MASK。

其他影响: 无

uint32 Comp_GetInterruptSourceMasked (void)

说明： 为CTBm模块中的所有比较器返回被中断掩码屏蔽的中断请求寄存器。
返回相应中断请求和掩码位之间按位AND运算后得到的结果。

参数： 无

返回值： 启用中断源的掩码。每个组件实例均有一个掩码值：Comp_INTR_MASKED。

其他影响: 无

void Comp_Sleep(void)

说明： 这是让组件准备进入睡眠状态的首选API。Sleep() API保存了当前组件的状态。在调用CySysPmDeepSleep()或CySysPmHibernate()函数前先调用Comp_Sleep()函数。“深度睡眠操作”选项会影响该函数的实现情况。

参数： 无

返回值： 无

其他影响： 无

void Comp_Wakeup(void)

说明： 此API函数是将组件恢复到调用Sleep()前的状态。Wakeup()函数会重新使能该组件。“深度睡眠操作”选项影响着该函数的实现情况。

参数： 无

返回值： 无

其他影响： 无

全局变量

函数	说明
Comp_initVar()	指示比较器是否已初始化。该变量初始时为0并在第一次调用Comp_Start()时设置为1。这允许第一次调用Comp_Start()子程序后组件无需重新初始化便可重新启动。 如果需要重新初始化组件，请在调用Comp_Start()之前先调用Comp_Init()。或者，可通过调用Comp_Init()和Comp_Enable()函数重新初始化比较器。

示例固件源代码

在 Find Example Project（查找示例项目）对话框中，PSoC Creator 提供了大量的示例项，包括原理图和示例代码。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要查看通用示例，请打开“Start Page”或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options**（滤波器选项）来限定可选的项目列表。

更多有关信息，请参考《PSoC Creator 帮助》中主题为“查找示例项目”的部分。

MISRA 合规性

本节介绍了 MISRA-C:2004 合规性和本器件的偏差情况。定义了下面两种类型的偏差：

- 项目偏差 — 适用于所有 PSoC Creator 组件的偏差

- 特定偏差 — 仅适用于该组件的偏差

本节提供了有关组件特定偏差的信息。*系统参考指南*中的“MISRA 合规性”章节，介绍了项目偏差以及有关 MISRA 合规性验证环境的信息。

比较器组件具有如下特定偏差：

规则	规则类	规则说明	偏差说明
19.7	建议	函数应优先于类函数宏使用。	因为使用函数宏以实现更高的代码效率而导致了偏差。

API 储存器的使用情况

根据编译器、器件、所使用的 API 数量以及组件的配置情况的不同，组件的存储器使用情况也不一样。下表显示了在指定组件配置中所有可用的 API 的存储器使用情况。

通过使用释放模式下相应的编译器，可以完成测量操作。在该模式下，存储器的大小得到优化。有关特定的设计，可分析编译器生成的映射文件以确定存储器使用情况。

PSoC 4 （GCC）

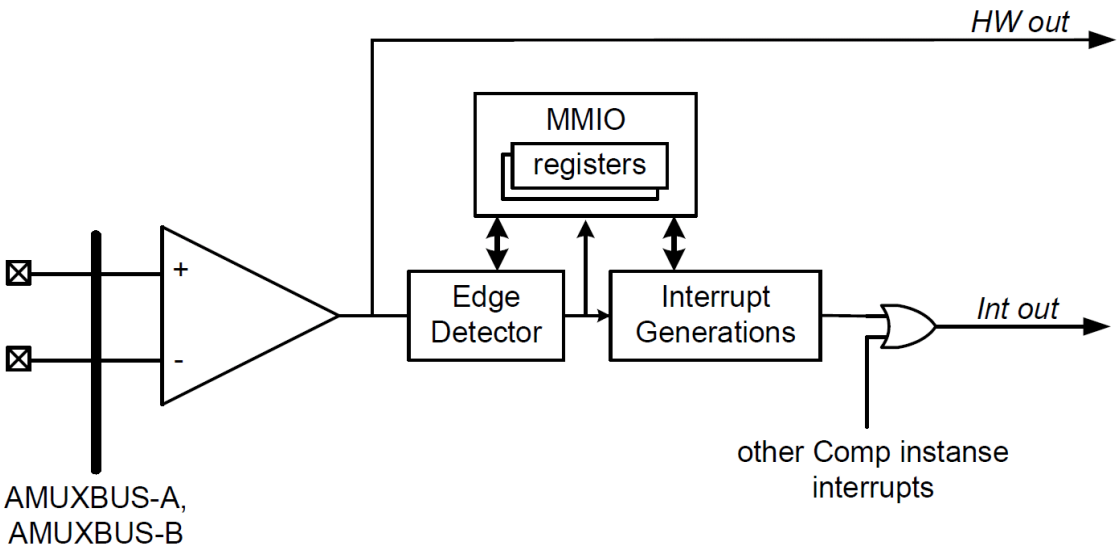
配置	PSoC 4100/PSoC 4200		PSoC 4200 BLE	
	闪存（字节）	SRAM（字节）	闪存（字节）	SRAM（字节）
默认（深度睡眠操作被禁用）	476	9	484	9
深度睡眠操作被使能	N/A	N/A	432	8

功能说明

PSoC 4 比较器组件将使用 CTBm 中配置为比较器的运算放大器。CTBm 支持 3 个功耗选择，可以用来平衡比较器的响应速率和电源使用率。

框图和配置

下面显示的是一个高级框图。



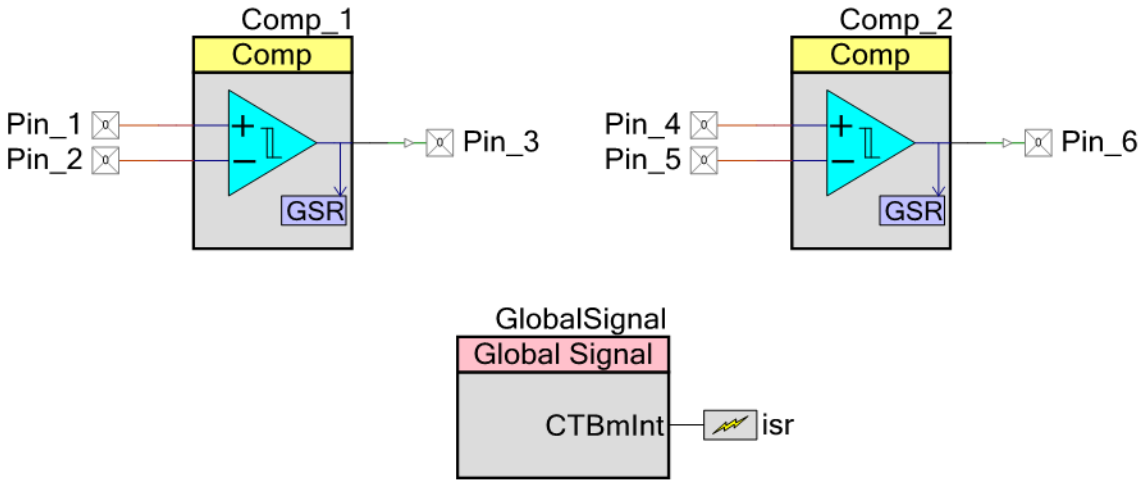
在低功耗模式下，发送和使用单独的比较器中断输出来唤醒系统前，应先对它们进行“OR”运算，使之成为单个异步中断源。单独比较器中断是被 INTR_MASK 屏蔽的。INTR_MASKED 寄存器将捕获屏蔽结果。将 INTR 寄存器位置 1 可清除该中断。有关每个寄存器的详细说明，请查看合适的器件*技术参考手册（TRM）*。

中断服务子程序

根据*中断输出模式*设置（上升沿、下降沿或双边沿），比较器组件为各种事件支持中断。任何已经使能的中断配置为 true 时，此中断信号将变为高电平。

注意：中断不会被自动清除。清除中断是用户的责任。通过将‘1’写入到相应中断寄存器位，可以清除该中断。使用 *Comp_ClearInterrupt(Comp_INTR)*或 *Comp_ClearInterruptOutput()* API 清除中断源是优选方法。

示例框图



建议使用下列代码：

```
CY_ISR(Comp_Interrupt)
{
if (Comp_1_GetInterruptSourceMasked() & Comp_1_INTR_MASKED)
    {
        /* Interrupt is not cleared automatically.
           It is user responsibility to do that.   */
        Comp_1_ClearInterrupt(Comp_1_INTR); /* or Comp_1_ClearInterruptOutput() */

        /* Add user interrupt code to manage interrupt. */
    }

/* This is alternative to (Comp_2_GetInterruptSourceMasked() &
   Comp_2_INTR_MASKED) */
if (Comp_2_GetInterruptOutputStatus())
    {
        /* Interrupt is not cleared automatically.
           It is user responsibility to do that.   */
        Comp_2_ClearInterrupt(Comp_2_INTR); /* or Comp_2_ClearInterruptOutput() */

        /* Add user interrupt code to manage interrupt.   */
    }
}
void main()
{
    Comp_1_Start();
    Comp_2_Start();
    isr_StartEx(Comp_Interrupt);
    CyGlobalIntEnable; /* Enable global interrupts. */

    for(;;)
    {
        /* Place your application code here. */
    }
}
```

在低功耗模式下工作

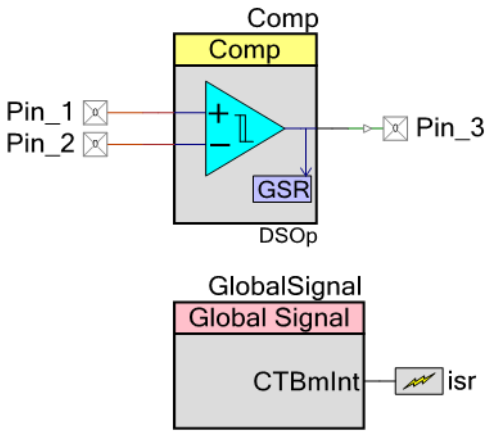
PSoC 4200 BLE 组件能与全局信号组件配合使用，从而能作为深度睡眠模式的唤醒源，具体如下所示。

根据前面讲述的内容创建原理图。应该选择“组合 CTBm 中断（CTBmInt）”作为全局信号参考组件中断的来源。每当使能比较器来生成中断时，该组件将触发输出。

在深度睡眠模式下，只能将比较器组件输入引脚分配给专用器件引脚。有关用于特定引脚连接的部件，请参考器件数据手册。

当连接到正端输入（Pin_1）的电压大于负端输入电压（Pin_2）时，会生成唤醒事件。

示例框图



建议使用下列代码：

```
CY_ISR(Comp_Interrupt)
{
    /* Interrupt is not cleared automatically.
       It is user responsibility to do that.   */
    Comp_ClearInterrupt(Comp_INTR); /* or Comp_ClearInterruptOutput() */

    /* Add user interrupt code to manage interrupt.  */
}

void main()
{
    Comp_Start();
    isr_StartEx(Comp_Interrupt);
    CyGlobalIntEnable;     /* Enable global interrupts. */
    Comp_SetInterruptMode(Comp_INTR_RISING);
    Comp_SetInterruptMask(Comp_INTR_MASK); /* or Comp_EnableInterruptOutput() */

    for(;;)
    {
```

```
            /* Place your application code here. */

            CySysPmDeepSleep(); /* Enter Deep Sleep mode*/

            /* Place your application code here. */
        }
    }
```

放置

每个比较器直接被连接到指定的 GPIO，其输入则被连接到内部结构。输出连接路由至数字结构。有关用于特定引脚连接的部件，请参考器件数据手册。

寄存器

有关寄存器的详细信息，请参见芯片的《技术参考手册》（TRM）。

组件调试窗口

通过 PSoC Creator，您可以查看设计中有关各组件的调试信息。每个组件窗口中列出了实例的存储器和寄存器。有关硬件寄存器的详细说明，请参考相应的器件技术参考手册。有关该组件所使用的 UDB 寄存器的详细说明，请参考本数据手册中介绍寄存器的内容。

要想打开 Component Debug（组件调试）窗口，请进行下述操作：

1. 请确保调试器处于运行模式或中断模式。
2. 从 **Debug** 菜单中依次选择 **Windows > Components…**。
3. 在 Component Window Selector 对话框中，选择需要查看的组件实例并点击 **OK**。

已选的 Component Debug 窗口将显示在调试器框架内。更多有关信息，请参考《PSoC Creator 帮助》中标题为“组件调试窗口”的内容。

使用资源

比较器可以使用 PSoC 4 CTBm 模块中的运算放大器（连续时间模块 mini）。可用的 CTBm 模块数量取决于被使用的器件。更多详细信息，请参见器件数据手册。如果已选择了反转输出选项，那么也可使用 UDB 阵列中的单个宏单元。

直流和交流电气特性

除非另有说明，否则这些规范的适用条件是：–40 °C ≤ T_A ≤ 85 °C 且 T_J ≤ 100 °C。除非另有说明，否则这些规范的适用范围为 1.71 V 到 5.5 V。

直流规范

参数	说明	条件	最小值	典型值	最大值	单位
I_{DD}	Opamp模块电流。空载。		–	–	–	–
I_{DD_HI}	功耗 = 高		–	1000	1300	µA
I_{DD_MED}	功耗 = 中		–	320	500	µA
I_{DD_LOW}	功耗 = 低		–	250	350	µA
V_{IN}	电荷泵打开，V_{DDA} ≥ 2.7 V		–0.05	–	VDDA – 0.2	V
V_{CM}	电荷泵打开，V_{DDA} ≥ 2.7 V		–0.05	–	VDDA – 0.2	V
V_{OS_TR}	偏移电压，校准后	高功耗模式	1	±0.5	1	mV
V_{OS_TR}	偏移电压，校准后	中等功耗模式	–	±1	–	mV
V_{OS_TR}	偏移电压，校准后	低功耗模式	–	±2	–	mV
$V_{OS_DR_TR}$	偏移电压漂移，校准后	高功耗模式	–10	±3	10	µV/C
$V_{OS_DR_TR}$	偏移电压漂移，校准后	中等功耗模式	–	±10	–	µV/C
$V_{OS_DR_TR}$	偏移电压漂移，校准后	低功耗模式	–	±10	–	µV/C
CMRR	DC	VDDD = 3.6 V	70	80	–	dB
CMRR	PSoC 4200 BLE系列的直流电	VDDD = 3.6 V，高功耗模式	65	70	–	dB
Vhyst_op	迟滞		–	10	–	mV

交流规范

参数	说明	条件	最小值	典型值	最大值	单位
Comp_mode	比较器模式；50 mV驱动，Trise = Tfall（近似值）		–	–	–	–
T_{PD1}	响应时间；功耗 = 高		–	150	–	nsec
T_{PD2}	响应时间；功耗 = 中		–	400	–	nsec
T_{PD3}	响应时间；功耗 = 低		–	1000	–	nsec
T_op_wake	从禁用到使能的时间，无外部RC电路。		–	300	–	µSec

组件更改

本节列出了该组件各版本中的主要更改内容。

版本	更新内容	更改原因/影响
1.10.b	编辑数据手册。	为PSoC 4200 BLE器件添加了CMRR char数据。
1.10.a	编辑数据手册。	更新说明文本，使之符合模板。
1.10	添加了各种函数以使能和控制中断。	提供了与PSoC 4低功耗比较器相似的功能。
	在General选项卡上添加了深度睡眠模式的参数，以便控制深度睡眠模式下组件的可用性。	对其进行了更新，使之支持PSoC 4200 BLE器件。
	更新了API存储器的使用情况以及MISRA合规性章节。	
	删除了SaveConfig()和RestoreConfig() API的参考，因为它们都是空的。	
1.0	新组件	

©赛普拉斯半导体公司，2014-2015。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了明确的书面协议，否则赛普拉斯产品不保证产品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®是赛普拉斯半导体公司的注册商标，PSoC Creator™和 Programmable System-on-Chip™是赛普拉斯半导体公司的商标。该处引用的所有其它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。除上述指定的用途外，未经赛普拉斯明确的书面许可，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：赛普拉斯不针对此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。赛普拉斯保留在不做出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中，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。

产品使用可能受适用于赛普拉斯软件许可协议的限制。