AN86233

PSoC® 4 和 PSoC 模拟协处理器的低功耗模式以及降低功耗技术

作者：Kannan Sadasivam、Max Kingsbury

相关项目：有

相关器件系列：PSoC 4100/4200/4100M/4200M/4200L 和 PSoC 模拟协处理器

软件版本：PSoC Creator 3.3 SP2 及更高版本

相关应用笔记：要想获取完整的列表，请点击此处。

想要获取本应用笔记的最新版本或相关项目文件，请访问 http://www.cypress.com/go/AN86233。

更多代码示例？我们明白。

如需寻找包含上百 PSoC 代码示例并有不断更新的网上资源，请浏览我们的代码示例网页。您还可以在此处观看 PSoC 4 视频库。

AN86233 说明了如何使用 PSoC® 4 和 PSoC 模拟协处理器低功耗模式和它的特性，使其运行于低功耗模式的同时，仍能保持基本功能。主要内容包括 5 种功耗模式、PSoC Creator™ 电源管理功能以及其他低功耗技术和注意事项。它还包含三个 PSoC Creator 示例项目，用于演示低功耗编程的不同方面。

目录

1 简介 2
2 功耗模式汇总 2
3 低功耗模式的详细信息 3
3.1 睡眠模式 3
3.2 深度睡眠模式 3
3.3 休眠模式 4
3.4 停止模式 5
4 功耗模式唤醒汇总 6
5 功耗降低技术 7
5.1 关闭未使用的组件 7
5.2 以较低速度运行组件 7
5.3 降低供电电压 8
5.4 使用 PSoC 器件控制电流路径 8
5.5 使用 DMA 传输数据 8
6 其他功耗模式中需要注意的事项 9
6.1 时钟 9
6.2 WDT 10
6.3 GPIO 11
6.4 深度睡眠和休眠模式下的电压调节器 11
6.5 调试接口 11
7 示例项目 12
7.1 PSoC 4 的项目 14
7.2 PSoC 模拟协处理器项目 16
8 硬件配置 17
8.1 CY8CKIT-042 套件（适用于 PSoC 4200） 17
8.2 CY8CKIT-049-42xx 套件（适用于 PSoC 4200） 18
8.3 CY8CKIT-043 套件（适用于 PSoC 4200M） 18
8.4 使用 DMM 测量电流 19
8.5 近似功耗 19
9 总结 19
10 相关应用笔记 19
全球销售和设计支持 22
产品 22
PSoC®解决方案 22
赛普拉斯开发者社区 22
技术支持 22

1 简介

通过 PSoC 4 和 PSoC 模拟协处理器的低功耗模式（尤其是与其他功耗节省特性和技术一起使用）可降低总功耗，同时保留其基本功能。本应用笔记描述了器件的低功耗模式，提供了有关在活动模式下节省功耗方法的信息，并讨论了其他低功耗模式下的注意事项。

本应用笔记假设您已经熟悉使用 PSoC Creator 来对 PSoC 进行开发应用。如果您尚不熟悉 PSoC 4 和 PSoC 模拟协处理器，请分别在 AN79953 — PSoC 4 入门和 AN211293 — PSoC 模拟协处理器入门中查找它们的介绍内容。如果您尚未了解 PSoC Creator，请参考 PSoC Creator 主页。更多参考资源，请参见相关应用笔记。

注意：除非另有说明，否则所谓‘PSoC’或‘器件’此后指的是 PSoC 4 和 PSoC 模拟协处理器器件。

2 功耗模式汇总

PSoC 具有 5 种工作功耗模式。按照功耗优化程度和功能的顺序，这些模式分别为：活动、睡眠、深度睡眠、休眠和停止模式。表 1 显示的是每种功耗模式下的典型电流和唤醒时间。

表 1. 功耗模式规范

功耗模式	电流范围（典型值）（Vdd = 3.3 V 至 5.0 V）	唤醒时间			
		PSoC 4100/4200	PSoC 4100M/4200M	PSoC 4200L	PSoC 模拟协处理器
活动	1.3 mA 至 14 mA	–	–	–	–
睡眠	1.0 mA 至 3 mA	0	0	0	0
深度睡眠	1.3 µA 至 15 µA	25 µs	25 µs	25 µs	35 µs
休眠	150 nA 至 1 µA	2 ms	0.7 ms	0.7 ms	不支持
停止	20 nA 至 80 nA	2 ms	2 ms	1.9 ms	不支持

注意：表 1 所显示的数值是典型值。而对于特定条件下的值，请参考器件系列数据手册。

不同低功耗模式的差别主要表示在 CPU 和外设的可用性、唤醒和复位源、功耗模式的切换行为以及功耗优化程度等方面。表 2 显示的是 PSoC 的资源以及它们在不同功耗模式下的可用性。

表 2. 资源在 PSoC 功耗模式下的可用性

子系统	活动模式	睡眠模式	深度睡眠模式	休眠模式	停止模式
CPU	启用	保持 [1]	保持	关闭	关闭
SRAM	启用	启用	保持	保持	关闭
高速外设（SPI、UART 等）	启用	启用	保持	关闭	关闭
通用数字模块（UDB）	启用	启用	保持 [4]	关闭 [5]	关闭
通用模拟模块（UAB）	启用	启用	保持 [4]	关闭	关闭
SPI 从设备和 I²C 从设备（基于 SCB）	启用	启用	启用	关闭	关闭
高速时钟（IMO、ECO 及 PLL）	启用	启用	关闭	关闭	关闭
32 kHz 的低速时钟（ILO 和 WCO）	启用	启用	启用	关闭	关闭
欠压检测	启用	启用	启用	启用	关闭
CTB/CTBm（运算放大器和比较器）	启用	启用	启用 [3]	关闭	关闭
ADC	启用	启用	关闭	关闭	关闭
低功耗电压比较器	启用	启用	启用	启用	关闭
GPIO（输出状态）	启用	启用	启用	启用	冻结 [2]

[1] 保持：外设的配置和状态得到保持。器件进入活动模式时，外设继续运行。

[2] 冻结：系统中所有 GPIO 的配置、模式和状态都被锁定。无法修改 GPIO 的状态，直到器件再次进入活动模式，并且引脚被解锁为止。

[3] 一些器件支持被称为“深度睡眠”的低功耗模式。

[4] 进入深度睡眠模式前通过调用组件的 `_Sleep()` 函数和通过退出深度睡眠模式后调用 `_WakeUP()` 函数来保持基于 UDB 的所有组件的状态。

[5] 当 PSoC 器件处于休眠状态时，它会通过复位事件来重新初始化所有组件。

3 低功耗模式的详细信息

本应用笔记中的 *“AN86233_低功耗模式”* 示例项目演示了 PSoC 的四种低功耗模式。

- 活动模式是正常的工作模式。在这种模式下，所有外设都可用，CPU 处于活动状态。
- 在睡眠模式下，除 CPU 外，所有外设均可用。
- 在深度睡眠模式下，将禁用 CPU、大多数外设和 MHz 时钟。
- 在休眠模式下，将禁用所有时钟，但仍保持逻辑状态。
- 在停止模式下，将暂停 CPU、时钟以及所有外设的运行，但仍会保持或冻结 GPIO 的各种状态。

3.1 睡眠模式

在睡眠模式下，PSoC 的 ARM® Cortex®-M0/M0+ CPU 不执行指令，会等待中断发生。SRAM 得到保持，但是 CPU 不能对它进行读取或写入操作。通过 DMA，可以访问支持 DMA 的部分。另外，其他外设和时钟都会继续运行。

3.1.1 睡眠模式的唤醒源

可使用器件内的任何中断源将器件从睡眠模式唤醒。所有外设均可保持活动状态，并生成中断。

3.1.2 睡眠模式的转换

通过调用 API 函数 `CySysPmSleep()`，可以进入睡眠模式。通过该函数，可以将器件配置为睡眠状态。不需要调用其他 API。

当触发某个中断时，器件将退出睡眠模式。退出睡眠模式后，PSoC 再次进入活动模式。睡眠唤醒源的配置只需使能它们的中断即可。

3.1.3 睡眠模式使用情况

当 ADC、CapSense®、数字通信等外设或其他资源需要保持活动状态，但不要求 CPU 活动时，应该使用睡眠模式。这样可以降低 ADC 转换和数字通信传输操作等事件中所消耗的电流。

3.2 深度睡眠模式

在深度睡眠模式下，高频率时钟和需要高频率时钟的外设都被禁用。高频时钟包括：内部主振荡器（IMO）、外部晶体振荡器（ECO），以及锁相环（PLL）。请注意，ECO 和 PLL 不适用于所有 PSoC 器件。

内部低速度振荡器（ILO）时钟仍保持活动状态，并能为看门狗定时器（可用作睡眠定时器来将系统从深度睡眠模式唤醒）提供时钟脉冲。一些 PSoC 器件还带有监视晶振（WCO），该晶振可在深度睡眠模式下工作。

I^2C 模块可以继续作为从设备工作，以监控 I^2C 总线，从而允许在 I^2C 地址匹配时唤醒器件。

3.2.1 深度睡眠模式唤醒源

I^2C 地址匹配、看门狗定时器、GPIO 中断、CTB/CTBm 比较器中断以及低功耗比较器中断均能将器件从深度睡眠唤醒。看门狗定时器模块包含了可被单独配置以生成中断、复位或两者的多个定时器。这样 WDT 便能够作为睡眠定时器使用。

3.2.2 深度睡眠模式的转换

通过调用 CySysPmDeepSleep() API 函数，可以进入深度睡眠模式。通过该函数，可将器件配置为深度睡眠模式。如果您不想关闭可在深度睡眠模式下工作的组件，从而节省功耗和/或保持其当前状态的话，那么便不需要调用其他任何 API 函数。在该情况下，从深度睡眠模式唤醒后，调用 CySysPmDeepSleep()和_WakeUp()前，要使用组件的指定 API _Sleep()来保持组件的当前状态。唤醒后，在进入深度睡眠模式和调用_Start()函数前，可根据您的应用加快调用_Stop()函数。

当触发中断时，器件将退出深度睡眠模式。退出深度睡眠模式后， PSoC 4 将再次进入活动模式。睡眠唤醒源的配置只需要使能它们的中断即可。

3.2.3 深度睡眠模式的使用情况

如果不需要使用 PSoC 的高性能模拟和数字外设，但仍需要通过看门狗定时器或 I^2C 地址匹配事件来定期唤醒器件，那么应采用深度睡眠模式。

通过常规唤醒间隔，可以定期使用处于活动模式的外设，例如：使用 ADC 来读取数据或扫描 CapSense 按键输入。

3.3 休眠模式

在休眠模式下，PSoC 内的所有时钟和同步外设都被禁用。引脚和低功耗比较器可以保持活动状态，并且 SRAM 和 UDB 寄存器的状态也得到保持。

注意： PSoC 模拟协处理器器件系列不支持休眠模式。

3.3.1 休眠模式的唤醒源

引脚和低功耗比较器中断可以将器件从休眠模式唤醒。每次退出休眠模式都会引起器件复位，但仍保持 SRAM 和某些寄存器的状态，从而可以检测唤醒复位的原因。虽然保持 SRAM，但默认复位会将所有变量初始化为其用户定义初始化状态或零。为了避免某个变量在从休眠模式唤醒后被重新初始化，请使用该变量定义中的“CY_NOINIT”属性。该属性仅适用于全局或静态变量。想要确保某个变量不被初始化，您代码应确定该复位是否由从休眠模式唤醒事件导致。复位完成并返回活动模式后，您可以调用 CySysPmGetResetReason() API 函数来确定复位是由休眠模式唤醒导致的，还是由任何其他原因引起的。下面的代码说明了如何定义一个未初始化的变量以及确定该复位是否由休眠唤醒导致。

```
/* Define non-initialized global variable */
int32 CY_NOINIT testVarNoInit;

/* Inside of main()    */
/* Initialize variable when PSoC is not reset from hibernation wakeup */
if( CySysPmGetResetReason() != CY_PM_RESET_REASON_WAKEUP_HIB )
{
    testVarNoInit = 0;
}
```

3.3.2 休眠模式转换

通过调用 CyPmHibernate() API 函数，可以进入休眠模式。通过该函数，可以将器件配置为休眠模式。不需要调用其他任何 API，这是因为退出休眠模式造成复位事件时，包含时钟的所有组件都被断电并被重新初始化。只有低功耗比较器仍能保持活动状态，用于引起从休眠模式唤醒现象。

当触发某个引脚或低功耗比较器中断时，器件将退出休眠模式。退出休眠模式后，PSoC 将复位。复位完成并返回活动模式后，您可以调用 CySysPmGetResetReason() API 函数来检测休眠唤醒复位的原因。使用组件 API 可以检测指定引脚或比较器中断，因为这些寄存器的状态均被保持。

虽然不必要，但是在进入休眠模式前，您可通过调用 CySysPmfreezeIo() API 函数来选择锁定所有 IO 单元（GPIO）。在引脚可以再次转换状态前，器件从休眠模式被唤醒后，应调用 CySysPmUnfreezeIo() API 函数。使用这些函数能够确保在复位过程中和复位后不会发生意外的 GPIO 转换。

3.3.3 休眠模式的使用情况

不需要定期唤醒，但器件消耗的电流小于 1 µA 时，应该使用休眠模式。该模式也可以用于对模拟或数字信号转换的唤醒，这种情况下所消耗的电流最小。请注意，在器件从休眠模式唤醒前，如果您的代码被组织成一个状态机，并且 CPU 可以开始从先前已知的状态中执行代码，那么可以有效地使用休眠模式。您必须使用状态变量定义中的 CY_NOINIT 属性，以确保状态变量从休眠状态唤醒后不会被重新初始化。通过调用 `CySysPmGetResetReason()` API 函数，可检测休眠唤醒复位（如以上章节所述），从而确保器件是由休眠模式唤醒导致复位的。

3.4 停止模式

PSoC 供电引脚未被断电时，停止模式下所消耗的电流最小。所有外设都被禁用，并且**不会**保持 SRAM 和寄存器的状态。器件引脚可能处于“冻结”状态并保持它们的驱动模式和逻辑状态。专用唤醒引脚（P0[7]）用于将器件从停止模式唤醒。

注意： PSoC 模拟协处理器器件系列不支持停止模式。

3.4.1 停止模式的唤醒源

专用唤醒引脚 P0[7]是停止模式下唯一可用的唤醒源。通过调用 `CySysPmSetWakeupPolarity()` API 函数，可以将它的输入唤醒极性设置为上升沿或下降沿。

3.4.2 停止模式转换

通过调用 `CyPmStop()` API 函数，可以进入停止模式。通过该函数，可以将器件配置为停止模式（包括冻结 I/O 状态）。如果您使用了专用的唤醒引脚，则再次进入停止模式前通过调用 `CySysPmSetWakeupPolarity()` API 函数来设置它的输入唤醒极性。

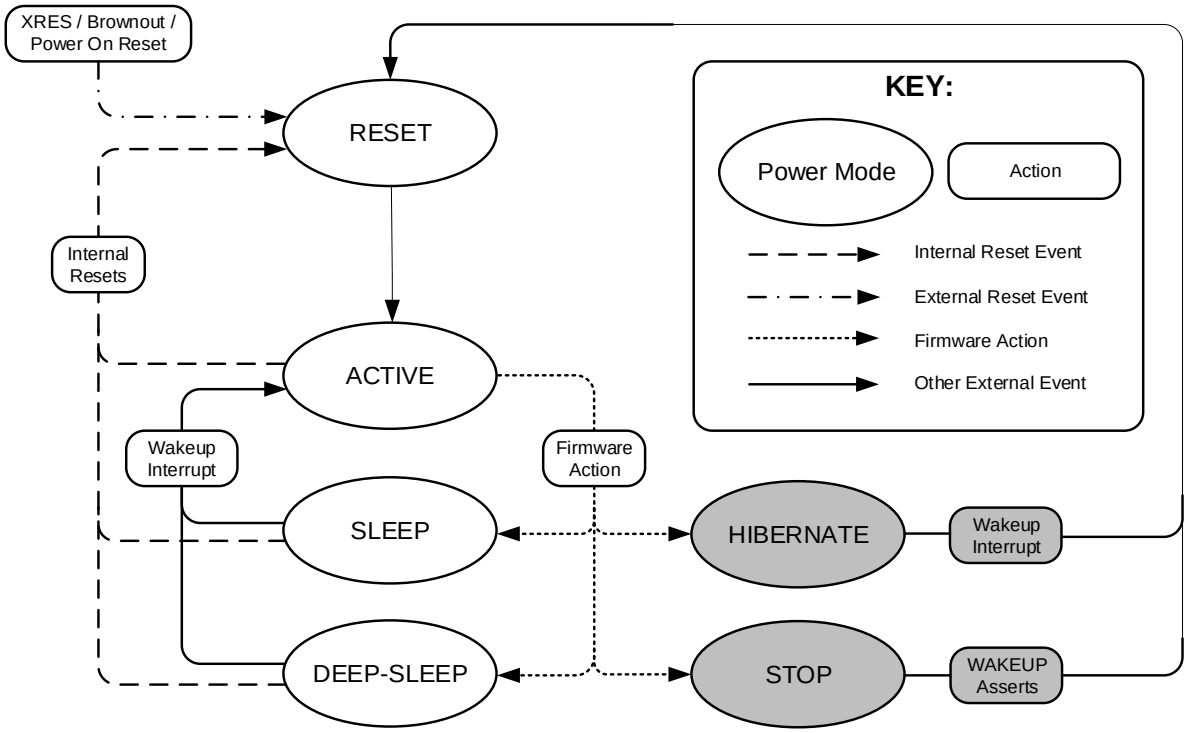
当触发专用唤醒引脚和复位信号处于低电平或循环供电时，将退出停止模式。退出停止模式时，PSoC 将复位。复位完成并返回活动模式后，通过调用 `CySysPmGetResetReason()` API 函数，可以确定该停止模式是通过切换唤醒引脚还是循环供电而退出。唤醒（引脚）复位后，GPIO 状态还会保持冻结状态，并在更改引脚状态前，必须通过 `CySysPmUnfreezeIo()` API 函数解除 GPIO 的冻结状态。由于 `CySysPmStop()`函数已经确保冻结了 IO 单元，因此进入停止模式前无需调用 `CySysPmUnfreezeIo()` API 函数。

3.4.3 停止模式的使用情况

需要最小的功耗和功能时，可以使用停止模式。在主机控制器或用户输入（如按键触摸）可以触发专用唤醒引脚，并且电源拓扑结构不允许断开器件电源的应用场合中，该模式非常有用。

图 1 显示了 PSoC 内功耗模式之间的切换。表 3 显示的是每种功耗模式下可用的唤醒源。

图 1. PSoC 功耗模式转换框图



注意： 灰色模块不适用于 PSoC 模拟协处理器

4 功耗模式唤醒汇总

有两个原因引起所有功耗模式退出，包括：循环供电和驱动复位输入为低电平。在**所有**功耗模式下，所有状态都被丢失。表 3 汇总了从当前低功耗模式退出并返回活动模式时所使用的各个事件和硬件。

表 3. 低功耗模式和唤醒源

低功耗模式	唤醒源	唤醒事件
睡眠	任意中断源	中断
	任意复位源	器件复位
深度睡眠	GPIO 中断	中断
	低功耗比较器	中断
	CTB/CTBm 比较器	中断
	SCB（I²C 地址匹配）	中断
	WDT[1]	中断或器件复位
	XRES（外部复位引脚）[2]	器件复位
休眠	GPIO 中断	器件复位
	低功耗比较器	器件复位
	XRES（外部复位引脚）[2]	器件复位
停止	WAKEUP 引脚（P0[7]）	器件复位
	XRES（外部复位引脚）[2]	器件复位

[1] 通过配置看门狗定时器，可在不同间隔产生中断或复位。有关可应用的 PSoC 器件系列的详细信息，请参见技术参考手册（TRM）。

[2] XRES 触发了一个完整的系统复位。所有状态（包括冻结 IO）均被丢失。在这种情况下，重新启动器件后便不能检测唤醒原因。

5 功耗降低技术

在许多应用中，通过适当地使用 PSoC 内部组件并对外部组件断电，您可以进一步降低消耗的电流。本节提供了一些用于降低功耗的技术。

5.1 关闭未使用的组件

在活动模式下降低功耗的最简单方式是关闭未使用的组件。

在活动或睡眠模式下所有可被禁用的组件在其 API 中都有一个 _Stop() 函数。该函数会立即停止组件的所有操作，并将其设置为最低功耗状态。组件可能正在执行某个任务，因此停止该组件前，必须检查它的状态。

```
/* <Check task status here.> */

/* Stop the Component. */
MyComponent_Stop();
```

通过调用某个组件的“Start”（启动)函数，可以重新启动它。

```
/* Start the Component. */
MyComponent_Start();
```

所有断电前必须保存其配置数据的组件，在其 API 中都有一个 _Sleep() 函数。_Sleep() 函数保存了所有必要的组件设置，然后调用 _Stop() 函数。在某些情况下，_Sleep() 函数只会调用_Stop() 函数。

```
/* < Check task status here.> */

/* Sleep the Component. */
MyComponent_Sleep();
```

当组件进入睡眠模式时，通过调用它的 _Wakeup() 函数可以唤醒它。该操作会恢复组件，使其进入睡眠模式前的状态。_Start() 函数也会使组件返回到工作状态，但是组件被重新初始化为其默认状态。

```
/* Wake the Component. */
MyComponent_Wakeup();
```

_Sleep() 和 _Stop() 函数都能够节省相同的功耗。它们的差别是组件是否需要从关闭时的准确位置恢复。

5.2 以较低速度运行组件

提高时钟频率时，时钟集成电路会消耗更大的电流。这是由于寄生和设计电容的充电和放电过程更迅速，需要的电流将更大。降低 PSoC 4 组件的工作频率可以大大降低电流消耗。该技术适用于 Cortex-M0/M0+ CPU、SAR ADC、数字组件以及其他组件。

5.3 降低供电电压

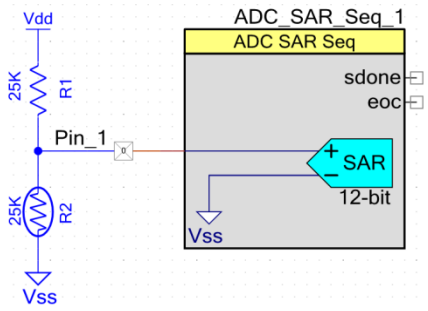
降低供电电压是减少总功耗的最简单方式。并且在电流值不变的情况下，则将供电电压从 5 V 降到 3 V，会使功耗降 40%。虽然 PSoC 器件可以在低于 1.8 V 的电压范围内工作，但仍要注意系统中其他器件的电压要求。

5.4 使用 PSoC 器件控制电流路径

您的 PCB 板上可能包含其他耗电组件，通过 PSoC 器件可以控制这些组件所消耗的电流。请注意，不能超过数据手册内所列出的最大引脚源电流和灌电流。

图 2 显示的热敏电阻应用是属于这种情况的良好示例。在这种情况下，PSoC 会通过使用模拟引脚上的电压（热敏电阻的电压随温度的变化而变化）来测量温度。

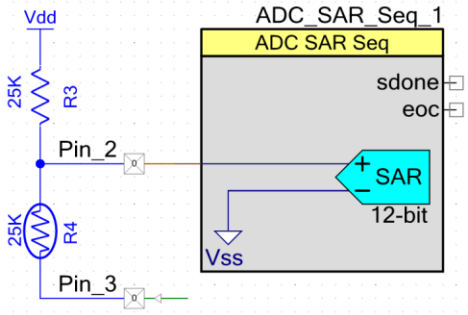
图 2. 典型热敏电阻应用



未使用 ADC 时，可以关闭它，但是外部组件还会消耗电能，这是因为仍要保留通过电阻器和热敏电阻的电流路径。PSoC 的一种简单解决方案是将第二个引脚作为接地开关使用，如图 3 所示。

在该配置中，通过将‘1’写入到 Pin_3 上可以停止电流消耗，并使引脚处于悬空状态。通过将两个电阻间的电压差降低为 0 V，可以消除电流消耗。将‘0’写入到 Pin_3 上，可持续消耗电流。这种省电技术仅仅使用了一个引脚和几行代码。

图 3．将 GPIO 作为接地开关使用



5.5 使用 DMA 传输数据

如果卸载 CPU 任务、停止 CPU 操作或使它并行处理其他任务，也可以节省功耗。PSoC 4200M、PSoC 4200L 和 PSoC 模拟协处理器器件都有一个 DMA 引擎，在活动或睡眠模式下可使用该引擎来传输数据（不需要使用 CPU）。

6 其他功耗模式中需要注意的事项

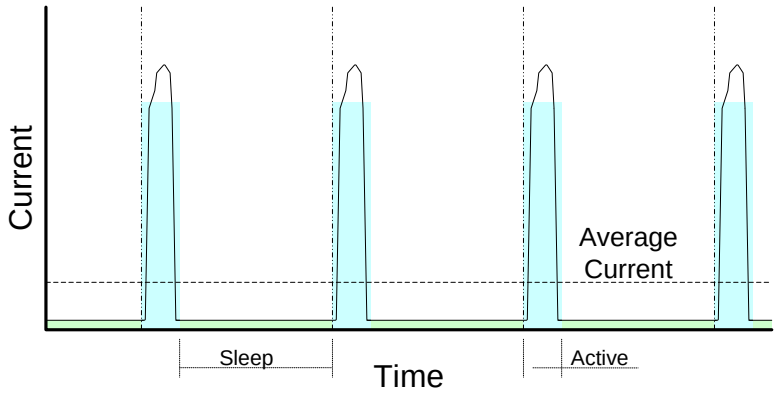
本节介绍了有关使用 PSoC 低功耗模式的各种提示、技巧以及推荐方法。

6.1 时钟

在某些情况下，使用频率更高的时钟可以降低平均电流消耗。例如，PSoC 需要每秒对传感器进行一次读取、执行几次计算操作，然后将结果发送给其他器件。

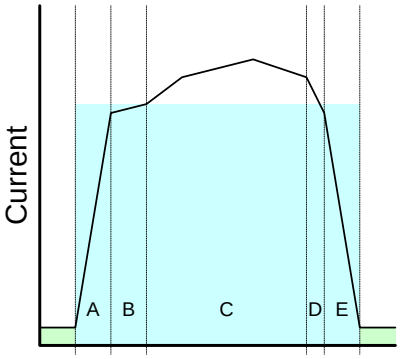
当 PSoC 处于空闲状态时，可以使用睡眠模式或深度睡眠模式来降低功耗，但是由于活动模式中的工作时间太久，因此平均电流消耗将更大。图 4 是该示例的电流消耗图示，其中系统时钟的频率为 3 MHz。

图 4. 时钟频率为 3 MHz 时的示例电流分析



根据 PSoC 唤醒时所执行的任务或计算，可以提高系统时钟执行的速度，从而更快地完成这些任务。这样可以降低平均电流消耗，因为 PSoC 器件处于活动模式的时间更短。图 5 是按任务划分的活动模式的时序图。

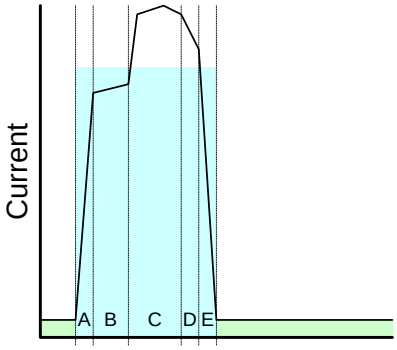
图 5. 工作频率为 3 MHz 时在活动模式下执行的任务分析



即使系统时钟频率上升，某些任务所需时间也不变，如传感器读取和数据传输。 但 CPU 执行的频率越高，其他任务所需的时间则越短。

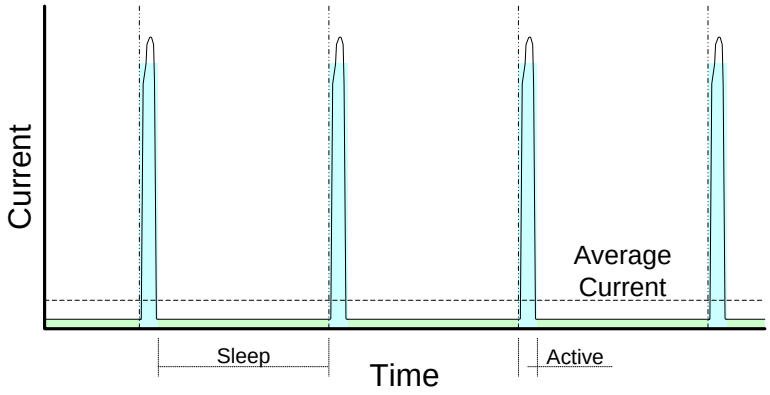
有时，虽然在活动模式下工作的时间短是一个优势，但是这会导致在高频率驱动时钟所消耗的功耗变得更大。假设最佳速度为 12 MHz，如图 6 所示。对于 12 MHz 时钟，在活动模式下的运行时间大概等于 3 MHz 时钟所需的时间的一半。图 7 显示当时钟频率更高时，峰值电流消耗更大，但是总体平均值却更低。

图 6. 工作频率为 12 MHz 时，在活动模式下执行的任务分析



A – Wake from sleep.
B – Read sensor data.
C – Manipulate data.
D – Transmit result.
E – Go back to sleep.

图 7. 时钟频率为 12 MHz 的示例电流分析



通过采用本应用笔记中的其它建议，您还可以降低峰值活动电流。项目 3：AN86233_DeepSleepADC 是一个示例项目，它每秒唤醒一次，使用 ADC 读取温度并通过 UART 传输数据，然后再返回深度睡眠模式。

6.2 WDT

看门狗定时器可以在活动、睡眠以及深度睡眠模式下执行。根据配置和工作条件，看门狗定时器内的多个计数器可以生成中断或复位。这样，看门狗定时器既可以保证操作的可靠性，又能代替传统睡眠定时器。

进入低功耗模式前，如果增加看门狗定时器的时间间隔或完全禁用它，则可以减少在活动模式下的运行时间，并降低总体功耗。看门狗定时器在休眠或停止模式下不能运行。

更多有关看门狗定时器的操作及有关 API 的信息，请参考 PSoC TRM 和 PSoC Creator 系统参考指南中的内容。

6.3 GPIO

当 PSoC 器件处于低功耗模式时，GPIO 可以继续驱动外部电路。当您需要在某个固定电平上保持外部逻辑时，该功能非常有用。但如果引脚不需要源或灌电流，这样便浪费功耗。

您应该分析系统设计，并确定 GPIO 在低功耗模式下的最佳状态。如果保持数字输出引脚为逻辑 1 或逻辑 0，那么可以使用引脚组件的 `_Write()` API 函数来设置它。

```
/* Set MyPin to '0' for low power. */
MyPin_Write(0);
```

将所有未使用的 GPIO 配置为模拟高阻状态，除非有特殊原因需要使用其他驱动模式。通过使用 `_SetDriveMode()` API 函数，可以设置引脚组件的端口级驱动模式。

```
/* Set MyPin to Alg HI-Z for low power. */
MyPin_SetDriveMode(MyPin_DM_ALG_HIZ);
```

PSoC 的灵活性使它易于管理 GPIO 驱动模式，从而避免不必要的漏电流。更多有关 GPIO 引脚配置的信息，请参见 AN86439，使用 GPIO 引脚。

在停止模式下，GPIO 驱动模式和数据寄存器可能处于“冻结”状态。通过调用 `CySysPmStop()` API 函数使 PSoC Creator 进入停止模式，引脚的状态将自动被冻结。唤醒复位后，通过调用 `CySysPmUnfreezeIo()` API 函数可以解除引脚的冻结状态，从而允许更改引脚的状态。

6.4 深度睡眠和休眠模式下的电压调节器

PSoC 有两个低功耗调节器，用于保持在深度睡眠和休眠模式下的逻辑状态。

- 深度睡眠调节器提供了足够大的电源，从而可进行快速唤醒，并能够满足子系统在深度睡眠模式下保持活动的要求。
- 在休眠模式下，休眠调节器所提供的电源只满足用于保持基本寄存器、存储器以及锁存器的逻辑状态。

这两个调节器均不给 V_{CCD} 提供电源。每个调节器提供了一个内部供电电压域，并且它未被连接到器件的引脚上。

注意： 休眠模式下的电压调节器不适用于 PSoC 模拟协处理器器件系列。这些器件不支持休眠模式。

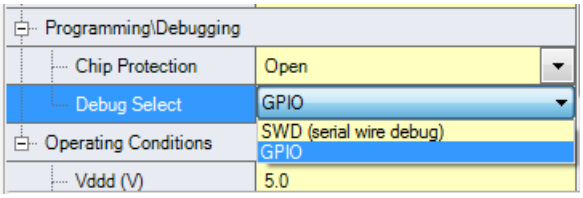
6.5 调试接口

PSoC 支持片上调试。在调试模式下，实际消耗的电流可能大于您的预期值。这种情况很正常，因为编程和调试接口在所有低功耗模式下均保持活动状态。

如果调试引脚被设置为 SWD 模式，并且连接了 MiniProg3 编程器/调试器，即使 PSoC 器件并不处于调试模式，但功率测量仍可能存在误差。

在所有出厂的芯片上，调试接口引脚均被设置为 GPIO 模式，但是新的 PSoC Creator 项目会默认将它们设置为 SWD 模式。仅可以在编程过程中更改控制着调试接口的寄存器。通过使用 PSoC Creator 项目 *.cydwr* 文件内的 System 选项卡，将引脚设置为 GPIO 模式，如图 8 所示。

图 8. 禁用调试接口从而降低功耗



当通过将这些引脚设置为 GPIO 模式来禁用调试接口时，必须经过复位后才能访问 PSoC 内的调试控制器。因此，不能将正在执行的项目连接到调试器上。如果这些引脚正处于 SWD 模式，为了尽可能降低功耗，在进入低功耗模式前需要执行以下某项操作：1）使用内部组件将这些引脚上拉或下拉；2）将 SWD 引脚转为高阻态。

建议将所有发行产品的调试接口引脚设置为 GPIO 模式。更多有关编程和调试的信息，请参见器件数据手册和 TRM 内容。

7 示例项目

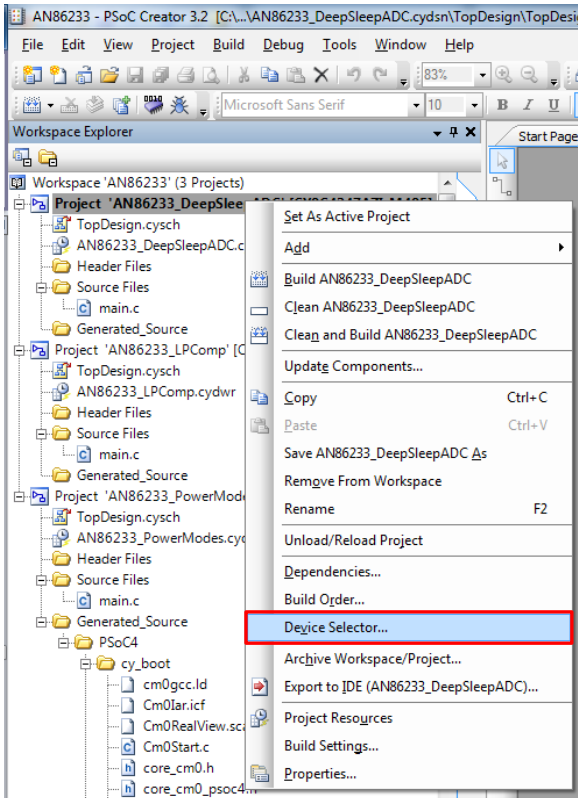
本应用笔记中所包含的 3 个 PSoC 4 系列器件项目和一个 PSoC 模拟协处理器对一些提及的概念进行了详细说明。每个示例的所有源代码均被保存在项目的 *main.c* 文件内，并有重要的注释。建议您抽出时间浏览每个示例的源代码，以了解基本操作。

PSoC 4 项目中默认设置的目标器件是 CY8C4245AXI-483，该器件使用于 CY8CKIT-042 和 CY8CKIT-049 套件，但是它也能适应于使用 4100/4200/4100M/4200M 器件的其他套件。这样，您可能需要改变连接到外部组件的引脚和器件选择。请查看特定的套件文档，了解有关的连接和器件类型。

请按照以下步骤配置其他任意 PSoC 器件的项目：

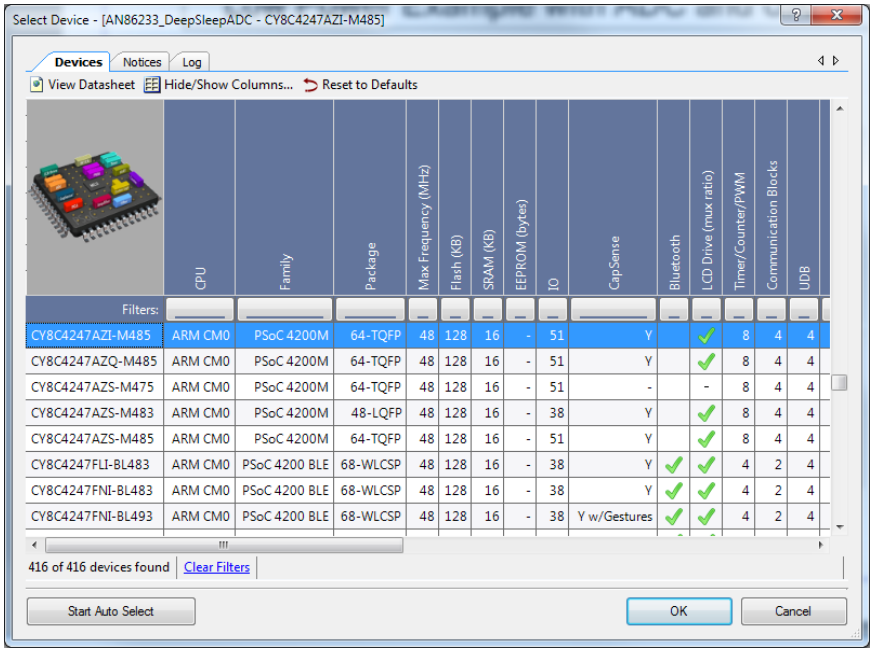
1. 在 PSoC Creator 中，打开项目。
2. 选择右键菜单中的 **Device Selector** 选项，如图 9 所示。

图 9. 选择 Device Selector 选项



3. 在弹出式菜单中选择合适的器件，如图 10 所示。

图 10. 器件选择器窗口



注意： 如果正在使用开发套件，请右击器件列表，并选择 **Select Default Device** 选项，然后选择合适的器件。

7.1 PSoC 4 的项目

7.1.1 项目 1：AN86233_PowerModes

通过设计该项目，很容易便能够将所有四种低功耗模式与活动模式进行比较。可以对某个位置上的代码进行编辑，从而将 PM_TO_DEMO 常量设置为所需的功耗模式常量。代码 1 显示的是一个选择深度睡眠模式的代码示例。

代码 1. 选择功耗模式

```
/* Power Modes Constants */
#define PM_SLEEP          1u
#define PM_DEEP_SLEEP     2u
#define PM_HIBERNATE      3u
#define PM_STOP           4u

/* Set PM_TO_DEMO to power mode */
#define PM_TO_DEMO  PM_DEEP_SLEEP
```

选择完成后，请编译项目并编程器件。该项目在活动模式下被启动，直到按下开发板上的开关为止。LED 会根据功耗模式常量的次数快速闪烁。在该实例中，LED 将在深度模式下闪烁两次。

该项目将处于低功耗模式，直到再次按下开关为止。然后，它会按相同的次数缓慢闪烁，以通知项目已返回到活动模式。通过电流表，可以评估每种模式下的电流。想要评估不同的功耗模式，请将 PM_TO_DEMO 设置为不同的功耗模式常量，重新编译项目，并重新编程 PSoC 器件。

7.1.2 项目 2：AN86233_LPComp

该项目演示了如何使用低功耗比较器（LPComp）来将 PSoC 器件从休眠模式唤醒，并返回到活动模式。为了完成该项目，需要在开发套件上添加两个 1 MΩ 外部电阻和一个电位器或电压源。这两个电阻构成了一个电压分频器，从而提供一个数值为 Vdd 一半的参考电压。电位器将模拟外部变化的电压，通过产生一个中断，使器件退出休眠模式，并返回到活动模式。电位器的可调整电压从大于参考电压转换成小于参考电压（或者反过来）时，会生成中断。

复位后，项目便处于活动模式。该项目处于活动模式时，LED 会被点亮；项目处于休眠模式时，LED 会关闭。按下开关，使 PSoC 器件进入休眠模式，直到 LPComp 检测到输入电压正在从大于参考电压的值转为小于参考电压的值（在该场合，电压为 Vdd/2），或反向转换。

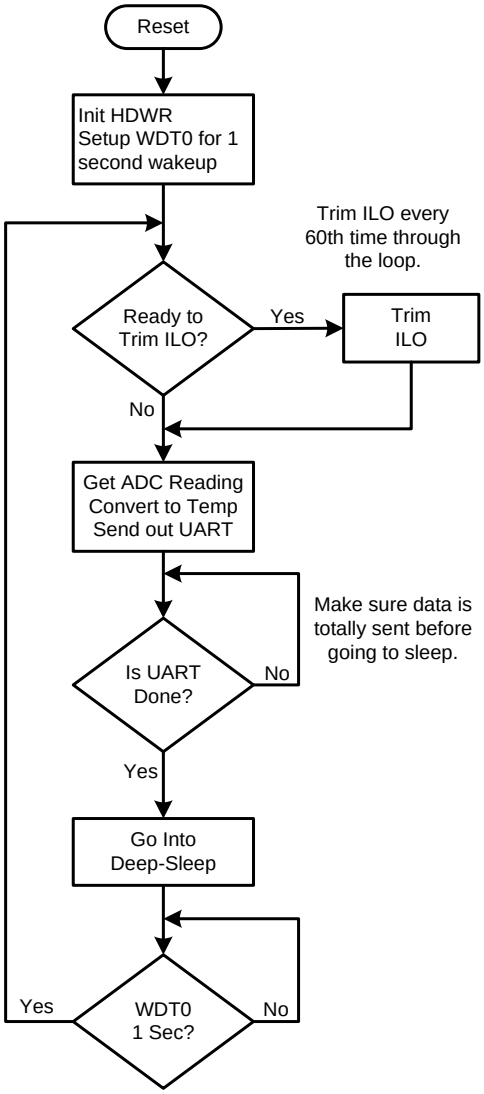
可以随意更改代码，所以按下开关时，器件便进入睡眠或深度睡眠模式，而不会进入休眠模式。在各种低功耗模式下，该操作应该是相同的，但所消耗的电能却不一样。

7.1.3 项目 3：AN86233_DeepSleepADC

该项目是一个示例，它演示了器件如何进行定期唤醒，通过 SAR ADC 读取数据，进行数据转换，通过串行接口（UART）传输数据，以及返回深度睡眠模式等内容。它每秒钟将执行一次该序列，但仍然保持各次间 99.9%的时间处于深度睡眠模式，使功耗降低三个数量级。

该项目使用了 ILO 及 WDT（WDT0），用于每秒从深度睡眠模式唤醒一次。ILO 是一个不很精确的低功耗振荡器，它的工作频率范围从 32 kHz –50%到+100%。而 IMO 的精确度更高（为±2%），它需要更大的功耗，并且不能在深度睡眠模式下工作。PSoC 器件处于活动模式时，IMO 一直工作。通过使用 IMO 以及 ILO Trim 组件，可以校准 ILO 的精确度。通过使用 ILO Trim，可以调整 WDT0 计数器周期，调整后的 WDT0 公差为±10%。ILO 的大漂移主要由电压和温度造成，因此不需要经常调整。在该示例中，组件从深度模式唤醒的每 60 次调整一次，或每分钟调整一次。你也可以在测量的 die 温度或者供电电压改变时进行调整而不是采用固定的 60 秒。请参见图 11，了解示例项目的程序流程。

图 11. AN86233_DeepSleepADC 程序流程

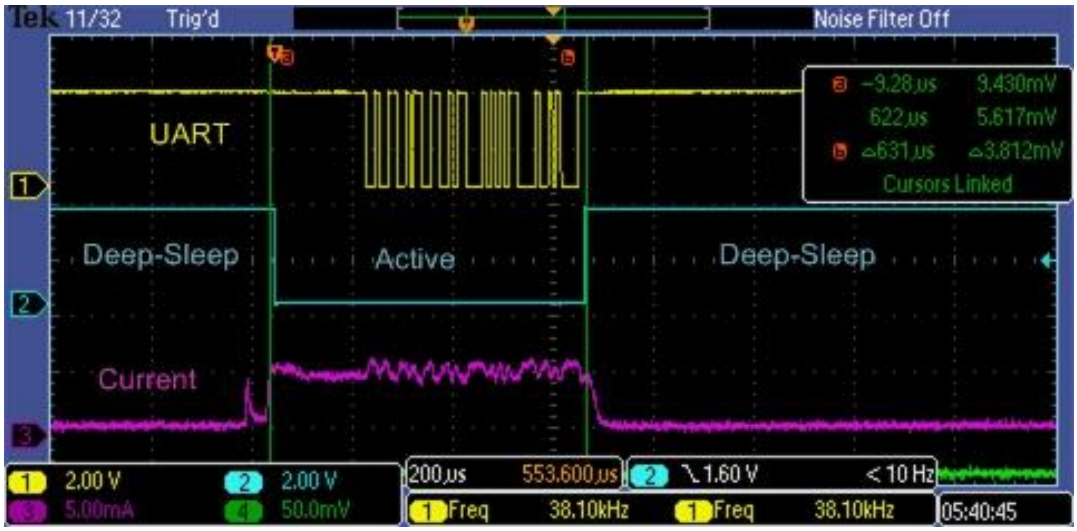


平均功耗

假设各种功耗模式间的占空比是相同的，为了确定该示例的平均功耗，需要测量每种模式下的时间和电流。固件的周期大约为一秒。因此，如果将 PSoC 器件处于活动模式的时间与处于深度睡眠模式的时间加起来构成一个周期，那么该周期大约为一秒。

首先，测量器件处于活动模式下的功耗和时间。图 12 显示了器件处于活动模式的时间约为 630 µs，并且它在这段时间内消耗大约为 5000 µA（5.0 mA）的电流。在图 12 中可以看到 UART 通信，并且要注意，PSoC 器件不会返回深度睡眠模式，直到 UART 完成它的传输为止。使用 LED 驱动为低电平有效信号的时长和测得的电流来确定器件处于活动模式还是深度睡眠模式。在深度睡眠模式下，电流为约 3.5 µA，传输信号的时间为约 999.4 ms。

图 12. 测量项目在活动模式下的参数



如果平均活动模式和深度睡眠模式的功耗，如表 4 所示，则可以容易看到通过使用深度睡眠模式，可以使活动模式的电流消耗 5 mA，从而平均值为 6.7 µA 并使功耗降低三个数量级。

表 4. 平均功耗计算

	深度睡眠模式	活动模式	总计	单位
时间（ms）	999.4	0.63	1000	ms
电流（µA）	3.5	5000		µA
时间 × I	3498	3150	6648	ms × µA
平均电流	6.7			µA

如果该项目由一个 300 mAh 的电池供电，并且不使用低功耗模式，那么电池寿命大约为 60 个小时（300 mAh / 5 mA）。通过使用功耗模式的转换方式，同样 300 mAh 的电池能够使用近 44,800 个小时或超过 5 年的时间。

当然，活动模式和深度睡眠模式下的占空比会极大地影响平均功耗，但每种功耗模式下的功耗都会影响平均功耗。两种电源模式间的转换使设计工作变得更加复杂，但更适合电池供电的应用。

7.2 PSoC 模拟协处理器项目

7.2.1 项目 1：AN86233_PowerModes

该项目与 PSoC 4 项目基本相同，但存在以下不同点：

- 休眠和停止模式不适用于 PSoC 模拟协处理器
- 在 PSoC 模拟协处理器中，将 IMO 和 SYCLK 频率设置为 24 MHz，而在 PSoC 4 中，将该值设置为 3 MHz。

8 硬件配置

赛普拉斯为 PSoC 系列提供了多款不同的开发套件，包括适用于 PSoC 4200 的 CY8CKIT-042、适用于 PSoC 4200M 的 CY8CKIT-044、适用于 PSoC 4200L 的 CY8CKIT-046 以及适用于 PSoC 模拟协处理器的 CY8CKIT-048。小外形低成本套件（被称为原型开发套件）也适用于 PSoC 系列，例如：适用于 PSoC 4200 的 CY8CKIT-049，适用于 PSoC 4200M 的 CY8CKIT-043。这些套件提供了不同的性能，但都包含一个连接到 PSoC 器件的用户开关和 LED。为了使器件正常工作，必须适当分配这些项目中每个套件的引脚。表 5 提供了 PSoC Pioneer 套件上的引脚映射，表 6 提供了 PSoC 4 原型开发套件上的引脚映射。

表 5. PSoC 4 Pioneer 套件上的引脚映射

功能	CY8CKIT-042（PSoC 4200）	CY8CKIT-044（PSoC 4200M）	CY8CKIT-046（PSoC 4200L）
红色 LED（低电平有效）	P1[6]	P0[6]	P5[2]
开关（低电平有效）	P0[7]	P0[7]	P0[7]
UART Tx*	P4[1]	P7[1]	P3[1]

*通过使用其他兼容引脚，也可连接到 UART Tx。在 CY8CKIT-044 和 CY8CKIT-046 套件上，该表中所列的引脚将被固定连接到板上 KitProg 的 USB 至 UART 桥接器。

表 6. PSoC 原型开发套件上的引脚映射

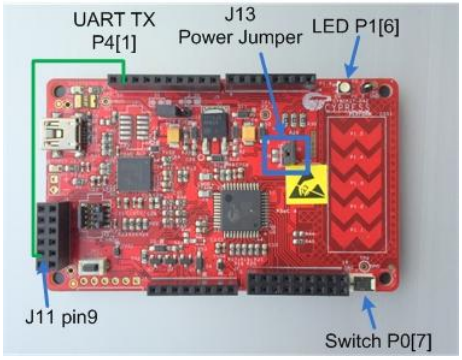
功能	CY8CKIT-049（PSoC 4200）	CY8CKIT-043（PSoC 4200M）
蓝色 LED（高电平有效）	P1[6]	P1[6]
开关（低电平有效）	P0[7]	P0[7]
UART Tx*	P4[1]	P7[1]

*通过使用其他兼容引脚，也可连接到 UART Tx。在 CY8CKIT-049 和 CY8CKIT-043 套件上，该表中所列的引脚被固定连接到板上的 USB 至 UART 桥接器。

8.1 CY8CKIT-042 套件（适用于 PSoC 4200）

该套件包含一个内置调试器和与多种通用 Arduino 扩展板相兼容的连接器。本应用笔记所述套件中用于测量电流的最关键性能便是电源跳线器 J13。为了测量 PSoC 器件的电流，应移除该跳线器，并在多个引脚间放置一个电流表。通过该操作，只能测量 PSoC 4 器件的电流，而不能测量电路板其他部分的电流。该电路板也包含了用于执行这些示例的 LED 和开关。另外，还可以将电路板作为 USB 转串口适配器使用。通过从 PSoC P4[1]到 J11 引脚 9 的连接，如图 13 所示，可以选择您喜欢的终端仿真器来监视 AN86233_DeepSleepADC 项目的串行输出。波特率为 115.2 kbps、8 位数据，并且无奇偶校验。更多有关使用 USB 转串口的性能的详细信息，请参见 PSoC 4 的 CY8CKIT-042 Pioneer 套件指南。

图 13. CY8CKIT-042 套件硬件

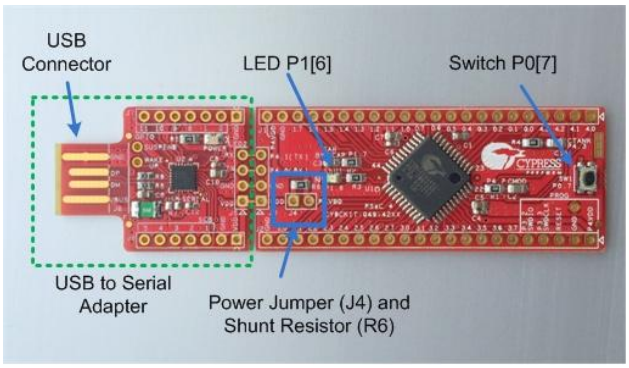


CY8CKIT-044 和 CY8CKIT-046 类似于 CY8CKIT-042，并且同样连接着开关、LED 和 UART Tx 线。更多有关这些套件的信息，请分别参见赛普拉斯网站上所提供的套件指南。

8.2 CY8CKIT-049-42xx 套件（适用于 PSoC 4200）

通过这些低成本套件，可以访问到所有 PSoC 4 引脚，并且这些套件还包含了与 CY8CKIT-042 套件相兼容的用户 LED 和开关。可通过电路板两侧其中一个“P4VDD”引脚或通过电路板尾部的 USB 连接器来提供电源。图 14 显示的是套件硬件。

图 14. CY8CKIT-049-42xx 套件硬件



该套件虽然没有内置调试器，但它包含了可用于引导加载项目的 USB 转串口适配器。首先要将可引导加载的组件添加到项目内，以便使用该接口。若有 PSoC MiniProg3 编程和调试套件（CY8CKIT-002），可以直接将它连接到用户安装的电路板右下角的连接器。有关如何使用 USB 转串口适配器来引导加载的更多信息，请下载赛普拉斯网站上的套件文档。

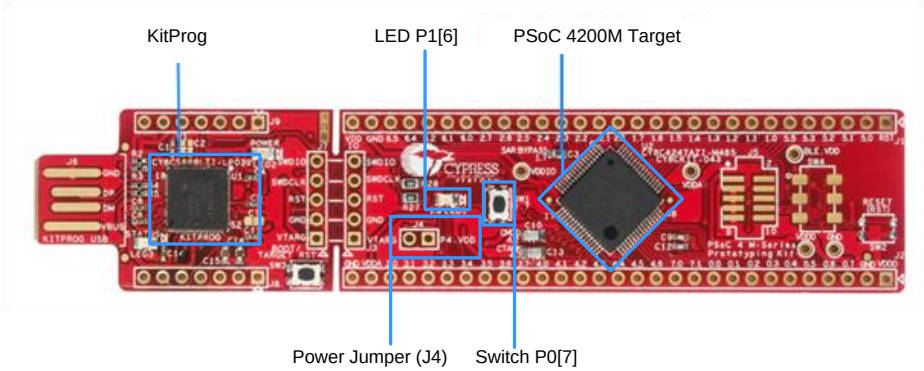
当 USB 端口给该套件供电时，要想通过该套件测量项目的电流，请移除零欧姆电阻 R6，并添加引脚或导线 J4，用于连接到电流表或数字万用表（DMM）上。

USB 转串口适配器提供了一个引导加载接口和一个串行端口适配器，用于显示示例项目 AN86233_DeepSleepADC 的输出。

8.3 CY8CKIT-043 套件（适用于 PSoC 4200M）

该套件包含一个内置调试器，并且具有与 CY8CKIT-049 相同的物理尺寸和形状。可通过电路板两侧其中一个“P4VDD”引脚或通过电路板尾部的 USB 连接器来提供电源。当 USB 端口为该套件供电时，要想通过该套件测量项目电流，请移除零欧姆电阻 R22，并将引脚或导线连接到（被连接到电流表或 DMM 的）J4 上。图 15 显示的是套件硬件。

图 15. CY8CKIT-043 硬件



该电路板也包含了用于执行这些示例的 LED 和开关。也可以将电路板作为 USB 转串口适配器使用。可以选择您喜欢的终端仿真器来监视 AN86233_DeepSleepADC 项目的串行输出，并且不用进行任何特殊的硬件连接。波特率为 115.2 kbps、8 位数据，并且无奇偶校验。更多有关使用 USB 转串行端口性能的详细信息，请参见 KitProg 用户指南。

8.4 使用 DMM 测量电流

使用 DMM 测量电流时，必须了解 DMM 中分流电阻的值在电流输入上，DMM 具有一个或多个（分流）电阻。这些电阻的大小为从不到 1 Ω到大于 10 kΩ。不同供应商的分流电阻甚至是同一家供应商的不同类型的分流电阻没有标准值。因为总是存在一个电压下降经过该分流，所以必须检查欧姆表手册，以了解分流电阻的值。这便意味着 PSoC 器件没有完全收到提供给它的电压。如果欧姆表中的分流电阻为 1 Ω或小于 1 Ω，那么测量 PSoC 电流时仅有几毫伏的压降，你可以忽略它。如果分流电阻的值为 1 kΩ，用来测量低电流，则 1 mA 的电流会引起 1 V 的压降。此外，更改 DMM 的测量范围时，请注意不要让 DMM 发生“先断后合”的现象，不然电源将被暂时断路，并且您的项目将被复位。

两个套件都有空间用于放置分流电阻，该方法摆脱对 DMM 分流电阻的依赖。CY8CKIT-049-42xx 具有一个零欧姆分流电阻（R6）。在该测量方法中，可以替换为一个小电阻，并且可以使用电压表来测量经过分流的电压。然后，很容易便能确定电流值。1 Ω和 100 Ω间的分流适用于大多数应用。CY8CKIT-042 中也存在接近 J13 的空间用于放置分流电阻（R6）。

8.5 近似功耗

器件数据手册和组件数据手册中提供了充足的信息，用于估算给定项目的功耗。为了简化该过程，这些数据手册中还提供了一个电子表格，该表格包含内部组件较大工作范围对功耗的典型要求。该电子表格 *PSoC4_Power_Estimator.xlsx* 位于 AN86233 页。由于各项目不同，所以该电子表格所提供的功耗计算只是一个估算值，但该估算值比较接近实际值，它足以能够在完成设计前提供良好的反馈。该电子表格中包含几个选项卡，请确保在输入数据前，您已经仔细阅读“Instructions”选项卡。

9 总结

好的想法和良好的设计之间可能因为功耗问题而存在差异。通过使用 PSoC 4 和 PSoC 模拟协处理器中的多个节省功耗性能，您可以优化设计，并确保功耗量最低。

10 相关应用笔记

这些应用笔记提供了更多有关本文档中未完整说明的主题的信息：

- AN79953 — PSoC 4 入门
- AN211293 — PSoC 模拟协处理器入门
- AN85951 — PSoC 4 CapSense 设计指南
- AN86439 — 使用 PSoC 4 的 GPIO 引脚
- AN90114 — PSoC 4000 系列低功耗系统设计技术
- AN77900 — PSoC 3 和 PSoC 5LP 的低功耗模式和降低功耗技术

关于作者

姓名： Kannan Sadasivam

职务： 应用工程师

背景： Kannan 已经获得模型工程学院（Model Engineering College）的电子与通信学士学位。他编写了许多关于混合信号设计及嵌入式系统的文章。

姓名： Max Kingsbury

职务： 高级应用工程师

背景： Max 已经获得美国华盛顿大学（Washington State University）的电气工程学士学位。他擅长编写高质量的技术文档和制作调试嵌入式设计。

文档修订记录

文档标题：AN86233 — PSoC® 4 和 PSoC 模拟协处理器的低功耗模式以及降低功耗技术

文档编号：001-89584

版本	ECN	变更者	提交日期	变更说明
**	4149169	LUFL	10/07/2013	本文档版本号为 Rev**，译自英文版 001-86233 Rev*A。
*A	4992956	LUFL	10/30/2015	本文档版本号为 Rev*A，译自英文版 001-86233 Rev*D。
*B	5255827	YLIU	05/10/2016	本文档版本号为 Rev*B，译自英文版 001-86233 Rev*F。 已更新为新模板。
*C	5495736	MEH	10/26/2016	没有技术更新。 完成日落评论。
*D	5815761	AESATMP9	07/14/2017	更新徽标和版权。

全球销售和设计支持

赛普拉斯公司具有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。要想查找离您最近的办事处，请访问赛普拉斯所在地。

产品

ARM® Cortex® 微控制器	cypress.com/arm
汽车级产品	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
物联网	cypress.com/iot
存储器	cypress.com/memory
微控制器	cypress.com/mcu
PSoC	cypress.com/psoc
电源管理 IC	cypress.com/pmic
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线连接	cypress.com/wireless

PSoC®解决方案

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

赛普拉斯开发者社区

论坛 | WICED IoT 论坛 | 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

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。

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

赛普拉斯半导体公司，2013-2017 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。