AN77900

PSoC® 3 和 PSoC 5LP 低功耗模式和功耗降低技术

作者： Greg Reynolds

相关器件系列：所有 PSoC 3 和 PSoC 5LP 器件

相关代码示例：有

软件版本：PSoC Creator™4.1 SP1 或更高版本

要获取相关应用手笔记的完整列表，请单击此处。

AN77900 介绍了 PSoC 3 和 PSoC 5LP 的低功耗模式和特性。主要内容包括 PSoC 功耗模式、电源管理 API 和寄存器、低功耗设计技术以及注意事项。相关的 PSoC Creator 示例项目演示了低功耗设计的方法。。

目录

1 简介 2
1.1 各种功耗模式及其转换 2
1.2 唤醒源 3
2 活动模式下降低功耗 4
2.1 关闭未使用的组件 4
2.2 以更慢的采样率使用 ADC 4
2.3 使用 PsoC 引脚控制电流路径....5
2.4 动态更改时钟速度 6
2.5 使用 DMA 传输数据....9
2.6 使用备用活动模式 9
2.7 使用功耗模式配置寄存器....9
3 其他功耗模式的注意事项 9
3.1 频率更高的时钟可能会降低功耗....9
3.2 32 kHz 晶体的低功耗模式 11
3.3 PSoC 睡眠模式下的低压检测（LVD）中断.....12
3.4 PSoC 睡眠模式下的 SegLCD 驱动器....12
3.5 PSoC 3 升压转换器的低低功耗模式....12
3.6 PSoC 5 LP 升压转换器的低功耗模式 13
3.7 快速 IMO 启动 13
3.8 PSoC 睡眠模式下的看门狗....14
3.9 PSoC 低功耗模式下的 GPIO....14
3.10 PSoC 低功耗模式下的 SIO....15
3.11 PSoC 低功耗模式下的数字模块 15
3.12 PSoC 低功耗模式下的参考源....16
3.13 睡眠和休眠模式下的电压调节器....17
3.14 芯片编程时的功耗 17
3.15 调试接口是否使能 17
4 近似功耗 18
5 总结 18
6 相关应用笔记 19
附录 A. 低功耗示例项目....20
A.1 两个低功耗演示项目....20
A.2 电压警报 — 无优化 21
A.3 电压警报 — 优化....22
附录 B. DVK 上的电源测量 23
B.1 对 CY8CKIT-001 进行修改....23
B.2 CY8CKIT-030 和 CY8CKIT-050 的修改 24
附录 C. 电源管理 API 和寄存器....26
C.1 CyPmSaveClocks()....26
C.2 CyPmRestoreClocks() 26
C.3 CyPmAltAct() 26
C.4 CyPmSleep() 26
C.5 CyPmHibernate()或 CyPmHibernateEx()....26
C.6 组件低功耗 API 26
C.7 寄存器的直接写操作....27
C.8 电源管理 API 流程图 28
C.9 电源管理 API 寄存器参考 29

1 简介

使用 PSoC 3 和 PSoC 5LP 的低功耗模式，可以在在满足功能的情况下降低设计的总功耗，特别是将低功耗模式与低功耗设计技术配合使用时效果更好。

本应用笔记介绍了 PSoC 低功耗模式的基本原理，提供了有关在活动模式下降低功耗方法的信息，并讨论了其他低功耗模式下的注意事项。本文档假设您已经熟悉了 PSoC 3 和 PSoC 5LP 器件的架构和 PSoC Creator。

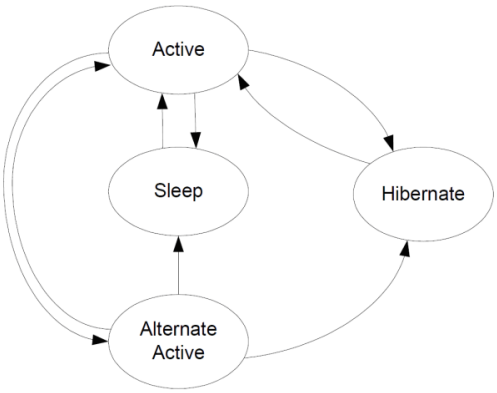
1.1 各种功耗模式及其转换

PSoC 3 和 PSoC 5LP 器件具有四种工作模式：活动模式、备用活动模式（AltAct）、睡眠模式和休眠模式。

- 活动模式是器件的主要工作模式，它是启动时的默认功耗模式。活动模式消耗的电源最多。
- 备用活动模式模式与活动模式非常类似。它是活动模式的备用电源配置。在该模式下 CPU 被禁用，芯片内部高低频时钟均保持工作状态，各外设的使能状态可配置，独立于 CPU 运行的纯硬件外设可正常工作，比如 Timer/Counter/PWM 以及基于 UDB 资源组件的硬件外设等等。由于 CPU 被禁用，功耗相比活动模式低。
- 睡眠模式几乎禁用了所有子系统，从而降低平均电流消耗（PSoC 3 的平均电流消耗为~1 µA 和 PSoC 5LP 的平均电流消耗为~2 µA）。PSoC 3 的最大唤醒时间为 15 µs，PSoC 5LP 的最大唤醒时间为 25 µs。
- 休眠模式禁用了绝对最小系统之外的所有资源，这样可以最大程度降低功耗（PSoC 3 的~200 nA 和 PSoC 5LP 的~300 nA）。PSoC 3 的最大唤醒时间为 100 µs，PSoC 5LP 的最大唤醒时间为 125 µs。

可以从活动和备用活动模式转换到任何其他模式。从睡眠和休眠模式唤醒后只计入活动模式，如图 1 所示。

图 1. PSoC 3 和 PSoC 5LP 的功耗模式转换



从一种功耗模式转换到另一种功耗模式会影响 PSoC 中所有子系统的功能。PSoC Creator 提供的 API 有助于简化和管理这些功耗模式的转换过程。

1.1.1 活动模式

任何有效唤醒或复位事件都会使 PSoC 器件返回到活动模式并使能 CPU。返回到活动模式的操作通常是自动发生的，因此不存在用于执行该转换的 API 函数。

退出活动模式的典型方法是调用低功耗模式的 API 函数。这些函数会准备 PSoC 器件以进入低功耗模式，并更新用于控制全局功耗模式设置的寄存器。更多详细信息，请参考电源管理 API 和寄存器。您不是必须通过调用 API 函数来退出活动模式，但强烈建议您这样操作。

1.1.2 备用活动模式

进入备用活动模式的典型方法是调用 API 函数 CyPmAltAct()。如果有任何挂起中断，PSoC 器件将立即返回到活动模式。

未屏蔽唤醒源产生一个中断时，PSoC 将自动返回到活动模式。有关可用的唤醒源列表，请参考表 1。如果在备用活动模式下 CPU 没有被禁用，通过转换到睡眠或休眠模式可以退出备用活动模式。要在备用活动模式期间使能 CPU，请参考 PSoC 3 寄存器技术参考手册的 PM_STBY_CFG0 寄存器部分。

1.1.3 睡眠模式

进入睡眠模式的典型方法是调用 API 函数 CyPmSleep()。如果有任何挂起中断，PSoC 器件将立即返回到活动模式。

由于 CPU 和大多数子系统停止工作，因此退出睡眠模式的唯一方法是通过复位或唤醒事件。有关可用的唤醒源列表，请参考第第 3 页上的表 1。

1.1.4 休眠模式

进入休眠模式的典型方法是调用 API 函数 CyPmHibernate()。如果当前任何端口中断控制单元（PICU）中断被挂起，那么 PSoC 器件将立即返回到活动模式。

唯一退出休眠模式的方法是通过一个被使能的唤醒源或硬件复位。

1.2 唤醒源

唤醒源分为三种：周期、异步和复位。

- 周期唤醒源包括中央时轮（CTW）、读秒脉冲（OPPS）和 LCD 定时器。实时时钟（RTC）和睡眠定时器组件使用这些定时器。
- 异步唤醒源包括升压转换器、比较器、I²C、LVI 和 PICU。
- 复位唤醒源包括外部复位（XRES）引脚和看门狗定时器（WDT）。

不同的低功耗模式支持部分唤醒源或所有唤醒源。表 1 显示的是每个功耗模式中可用的唤醒源。

1.2.1 多个唤醒源

PSoC 应用可以使用多个唤醒源。例如，当按下某个按键（PICU）或外部温度过高（比较器）时，PSoC 器件需要定期被唤醒，以检查电池状态（OPPS）。

要想配置多个唤醒源，通过将多个参数通过 OR（或）运算结合起来调用功耗模式 API 函数。唤醒后，您必须读取中断状态寄存器，以识别唤醒源。更多详细信息，请参考电源管理 API 和寄存器。

表 1. 低功耗模式和唤醒源

唤醒源	PSoC 3			PSoC 5LP			注释
	备用活动	睡眠	休眠	备用活动	睡眠	休眠	
中断	✓			✓			必须使用中断组件。
CTW	✓	✓					可以配置睡眠时间。
睡眠定时器	✓	✓		✓	✓		睡眠定时器使用了 CTW。
OPPS	✓	✓					需要 32 kHz 的晶体。
RTC	✓	✓		✓	✓		RTC 使用 OPPS。
FTW	✓			✓			
PICU	✓	✓	✓	✓	✓	✓	所有未屏蔽引脚中断。
比较器	✓	✓	✓	✓	✓	✓	
I²C 地址匹配	✓	✓		✓	✓		固定模块从设备地址匹配。
段式 LCD 刷新	✓	✓[1]		✓	✓[1]		周期大小取决于设置。
升压转换器	✓	✓[2]	[3]	✓	✓	[3]	
WDT	✓	✓		✓	✓		如果未给 WDT 通知，它会生成一个信号以便复位系统。
LVI	✓	✓		✓	✓		
XRES	✓	✓	✓	✓	✓	✓	器件唤醒和复位。

1. LCD 低功耗特性不能同时与 CyPmSaveClocks()函数一起使用。
2. 如果 PSoC 3 处于睡眠模式，升压转换器可以在活动模式或待机模式中运行。建议使用待机模式。
3. 使用升压器时，不建议使用休眠模式，请使用睡眠模式。

2 活动模式下降低功耗

您的应用可能无法使用低功耗模式或者必须大部分时间工作在活动模式。您不需要进入低功耗模式仍可以降低活动模式下的平均功耗。

2.1 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。

在活动模式下可禁用的所有组件自身都有 API _Stop()函数。该函数会立即停止组件执行的所有操作，并设置为它的最低功耗状态。组件可能正在执行某个任务，因此在停止该组件前，必须检查它的状态。

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

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

组件停止后，通过调用 Start 函数可以重新启动它。

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

断电前必须保存其配置数据的所有组件，其自身都有 API _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 函数仅调用 Stop 函数。如果代码执行时间非常重要，那么请查看所生成的源代码以确认您是否能调用 Stop 函数代替。

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

/* Sleep the component. */
MyComponent_Sleep();

/* <Do something else here.> */

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

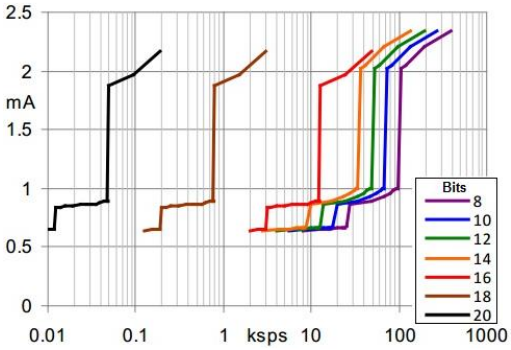
当组件进入睡眠模式时，通过调用它的 Wakeup 函数可以再次唤醒它。该函数会使组件恢复到其进入睡眠模式前的状态。Start 函数也会使组件返回到操作状态，但组件被初始化为默认状态。

Sleep 函数和 Stop 函数能够节省相同的功耗。它们之间的区别是是否会在关闭组件之前保存组件的当前设置，以确保组件可以从当前设置恢复工作。与本应用笔记相关的示例项目显示了如何使用 Stop/Start 和 Sleep/Wakeup 函数。

2.2 以更慢的采样率使用 ADC

PSoC 中的 ADC 存在一些不同的电源配置文件。各种配置文件会根据分辨率和采样率自动被使能。例如，当您切换电源模式时，PSoC 3 中 DelSig ADC 的功耗大量变化，如图 2 所示。

图 2. PSoC 3 DelSig IDD 与采样率（已缓冲）



使用稍慢的采样率可明显降低功耗。这时，如果将 ADC 配置为 16 位分辨率，则 10 ksps 和 12 ksps 之间的差值大约为 1 mA。

如果需要考虑活动模式下的功耗，那么应该检查组件数据手册以确定较慢的采样率是否能够明显降低功耗。

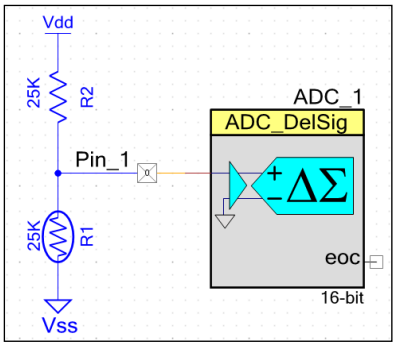
更多有关选择合适采样率和分辨率的信息，请参考 Delta-Sigma ADC 组件数据手册、PSoC 3 或 PSoC 5LP 架构技术参考手册（TRM）中介绍的内容。

2.3 使用 PsoC 引脚控制电流路径

您的 PCB 上可能有其他组件耗电，通过使用 PSoC 器件可以控制这些组件中的电流。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。

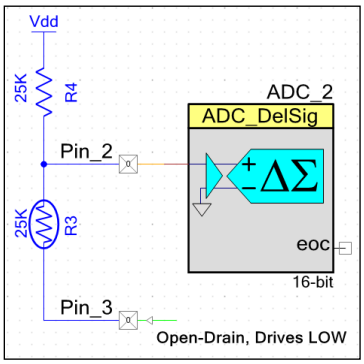
图 3 内显示的热敏电阻应用是属于这种情况的良好示例。在这种情况下，PSoC 会通过使用模拟引脚上的电压（热敏电阻的电压随温度的变化而变化）来测量温度。

图 3. 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，因为电阻器和热敏电阻中的电流没有被停止。PSoC 的一种简单解决方案是使用第二个引脚作为接地开关，如图 4 所示。

图 4. 使用 GPIO 作为接地开关



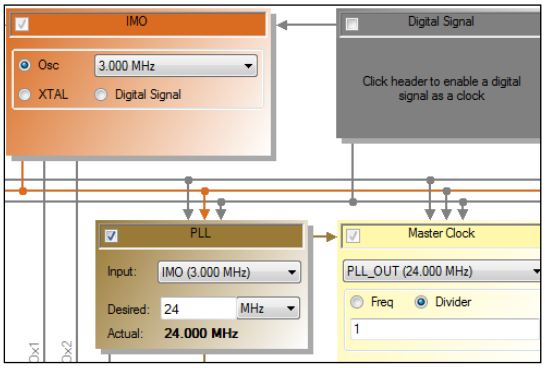
在该配置中，通过将‘1’写入到 Pin_3 上可以停止电流循环。写入‘0’便恢复电流循环。这种省电技术仅使用了一个引脚和几行代码。

2.4 动态更改时钟速度

PSoC 3 和 PSoC 5LP 能够在运行期间更改时钟速度。这样您可以在大部分时间内将这些时钟的速度设置为更慢，并且在需要进行复杂操作时增大它们的速度。

在新的项目中，默认时钟设置为 3 MHz 内部主振荡器（IMO），该 IMO 用于为一个 24 MHz 的 PLL 提供时钟，如图 5 所示。

图 5. 新项目的默认 IMO 和 PLL 设置



如果禁用了 PLL 并将 IMO 设置为 3 MHz，便可以降低电流消耗。这对空项目来说很好，但您的应用需要的时钟速度更快。

例如，如果将 16 位 ADC 的多采样模式设置为 1 万次采样/秒，那么主设备时钟的运行频率至少要 12 MHz，如图 6 所示。为了实现该操作，您可以将 IMO 的运行频率设置为 12 MHz 或将 PLL 的最小频率设置为 24 MHz。

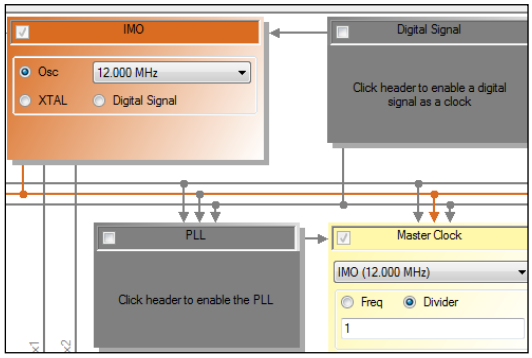
图 6. 警报 — ADC 设置的时钟太慢

Name	Desired Frequency	Nominal Frequency	Source Clock
USB_CLK	48.000 MHz	? MHz	IMOx2
Digital_Signal	? MHz	? MHz	
XTAL_32KHZ	32.768 kHz	? MHz	
XTAL	24.000 MHz	? MHz	
PLL_OUT	24.000 MHz	? MHz	IMO
ILO	? MHz	1.000 kHz	
BUS_CLK (CPU)	? MHz	3.000 MHz	MASTER_CLK
MASTER_CLK	? MHz	3.000 MHz	IMO
IMO	3.000 MHz	3.000 MHz	
Clock_1	1.000 kHz	1.000 kHz	Auto: IMO
ADC_DelSig_1_theACLK ⚠	2.590 MHz	3.000 MHz	Auto: MASTER_CLK
ADC_DelSig_1_Ext_CP_Clk ⚠	10.360 MHz	3.000 MHz	Auto: MASTER_CLK

当 ADC 未进行采样时，可以将该项目的 IMO 设置为 3 MHz。因此您只需要禁用 PLL 并将 IMO 在 3 MHz 和 12 MHz 之间进行切换。

PsoC Creator 需要您配置各种时钟，以支持您项目中所有组件的速度。在该示例中，您必须将默认配置设置为 12 MHz，如图 7 所示。

图 7. 动态更改时钟设置



完成该设计中所有设置操作后，您可以写入到固件来更改时钟速度。该示例使用了三个 API 函数来准确配置 PSoC：

- CyIMO_SetFreq() — 该函数设置了 IMO 时钟的频率。
- CyFlash_SetWaitCycles() — 该函数计算和设置等待周期数量，从而确保 CPU 能够正常进行闪存读写操作。
- CyDelayFreq() — 该函数计算和设置准确定时 CyDelay 操作所需要的周期数量。

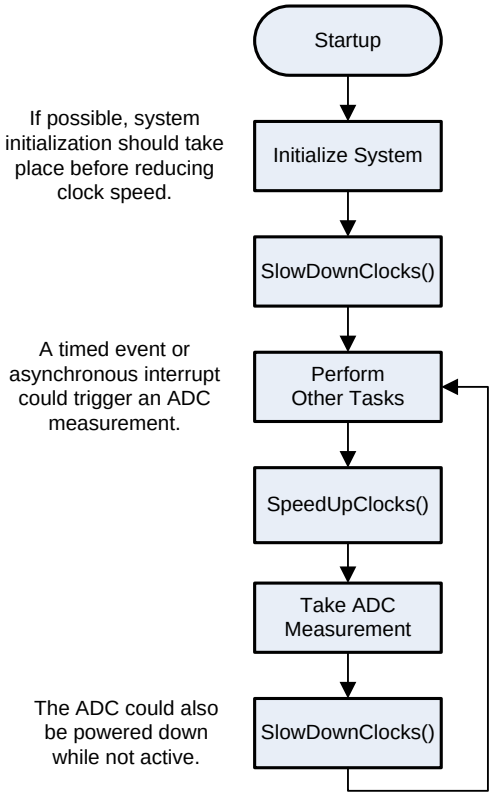
其他时钟没有自动被更改。您必须通过添加代码来调整它们的设置。有关用于调整这些设置的 API 和寄存器的信息，请参考组件数据手册和系统参考指南中介绍的内容。

您可以通过创建两个函数来更改 IMO 速度：

```
void SlowDownClocks(void)
{
    /* Set IMO frequency to 3MHz. */
    CyIMO_SetFreq(CY_IMO_FREQ_3MHZ);
    /* Set Flash wait to 3MHz. */
    CyFlash_SetWaitCycles(3);
    /* Set CyDelay frequency to 3MHz. */
    CyDelayFreq(3000000);
    /* Change any other active clocks. */
    OtherClock_SetDivider(0); /* 3MHz/1 */
}
void SpeedUpClocks(void)
{
    /* Set IMO frequency to 12MHz. */
    CyIMO_SetFreq(CY_IMO_FREQ_12MHZ);
    /* Set Flash wait to 12MHz. */
    CyFlash_SetWaitCycles(12);
    /* Set CyDelay frequency to 12MHz. */
    CyDelayFreq(12000000);
    /* Change any other active clocks. */
    OtherClock_SetDivider(3);/* 12MHz/4 */
}
```

在该示例中，系统被初始化后，时钟频率能够立即降低到 3 MHz。仅在 ADC 需要进行采样时，该频率才会递增到 12 MHz，如图 8 所示。

图 8. 时钟更改示例的流程图



时钟速度调整功能可以与其他节能技术一起使用，用于降低平均电流消耗（不需要使用低功耗模式）。

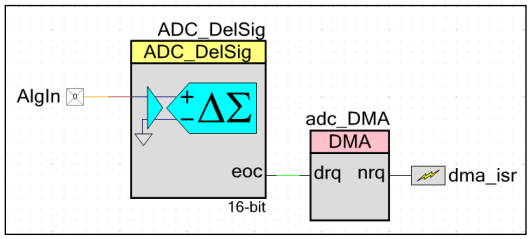
2.5 使用 DMA 传输数据

如果卸载 CPU 任务、停止 CPU 操作或使它并行处理其他任务，那么可以节省功耗。PSoC 3 和 PSoC 5LP 具有一个 DMA 引擎，可以在活动或备用活动模式下使用该引擎来传输数据（不需要 CPU 的干预）。

在图 9 中的示例内，转换过程完成时，ADC 将触发 DMA 传输。DMA 引擎将 ADC 结果转移到另一个位置（无需使用 CPU），然后它会触发一个中断来表明该传输已经完成。

DMA 使用的主题较复杂，所以本文档没有详细说明该主题。您可以在 www.cypress.com 网页上找到更多信息，包括各示例项目和应用笔记，如 AN52705 - PSoC® 3 和 PSoC 5LP — DMA 快速入门。

图 9. ADC 完成后触发的 DMA



2.6 使用备用活动模式

备用活动模式拥有一套私有的子系统控制寄存器集，可以用来控制各个功能模块在该模式下是否启动或禁用，包括 CPU。比如通过控制这些寄存器位可以在备用活动模式下禁用 CPU 并使能某些硬件外设，例如基于 UDB 的纯硬件外设等。在备用活动模式下，如果产生任何中断，器件将自动转换为活动模式并开始在活动模式下执行固件。这样比打开和关闭各个块更有效和更快。

2.7 使用功耗模式配置寄存器

在活动或备用活动模式下，您可以使用功耗模式配置寄存器来控制大多数子系统的电源。在活动模式下，14 PM_ACT_CFGx 寄存器用于控制电源和时钟。14 PM_STBY_CFGx 寄存器适用于备用活动模式。

根据项目设置，PSoC Creator 可生成用于功耗模式配置寄存器的一组默认值。这些值被存储在 *cyfitter_cfg.c* 文件内，并且在启动时被加载到寄存器内。通过写入到任意 CY_PM_ACT_CFGx 寄存器的操作，您可以在运行时间内更改该配置。

新的设置立即生效，但复位后，它将失效。另外，子系统中的中断会自动将这些寄存器中的某些位设置为‘1’。

最好直接调用组件提供的 Stop API 函数，因为它始终被映射到正确的物理子系统内。PSoC 3 和 PSoC 5LP 寄存器技术参考手册介绍了功耗模式配置寄存器的位图。

3 其他功耗模式的注意事项

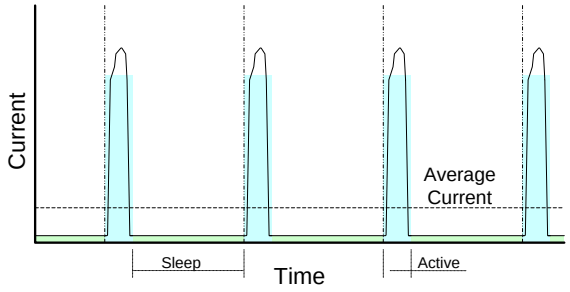
本节讨论了有关使用 PSoC 3 和 PSoC 5LP 低功耗模式的各种提示、技巧和建议方法。

3.1 频率更高的时钟可能会降低功耗

在某些情况下，使用频率更高的时钟确实可以降低平均电流消耗。例如，PSoC 需要每秒进行一次读取传感器、执行解析和计算操作，然后将结果发送给其他器件。

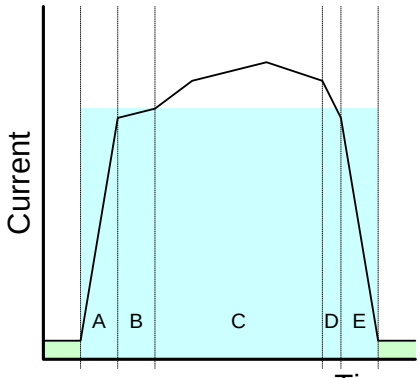
当 PSoC 处于空闲状态时，您可以使用睡眠模式来降低功耗，但是由于活动模式的工作时间，平均电流消耗将更大。图 10 是该示例的电流消耗图示，其中系统时钟频率为 3 MHz。

图 10. 时钟频率为 3 MHz 时的示例电流分析



如果您要查看 PSoC 唤醒时所执行的任务，通过系统时钟执行的速度越快，完成时间则越早。这样可以降低消耗的平均电流，因为 PSoC 器件在活动模式下的时间更短。图 11 是按任务划分的活动模式时序图。

图 11. 在工作频率为 3 MHz 的活动模式下执行的任务分析

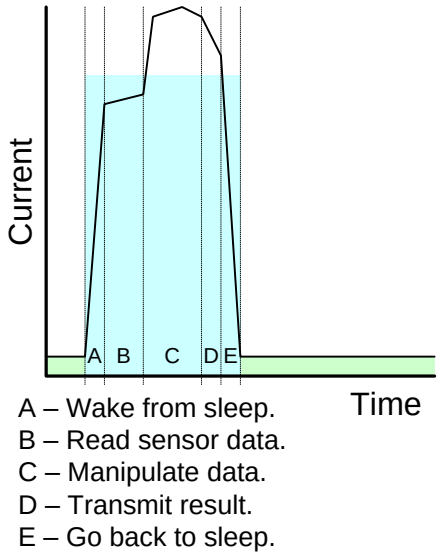


A – Wake from sleep.
B – Read sensor data.
C – Manipulate data.
D – Transmit result.
E – Go back to sleep.

即使系统时钟频率上升，某些任务所需时间也不变。传感器读取和数据传输是一个例子。但如果 CPU 执行的速率越快，其他任务需要的时间也越短。

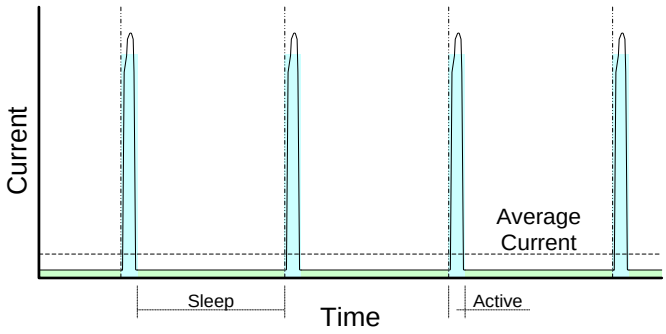
有时，虽然在活动模式下执行时间很短，但是运行时钟频率更高，功耗会很大。假设最佳速度为 12 MHz，如图 12 所示。

图 12. 工作频率为 12 MHz 时，在活动模式下执行的任务分析



在活动模式下的运行时间大概等于系统时钟频率较慢时所需运行时间的一半。图 13 显示的是当时钟频率更快时，峰值电流消耗更大，但总体平均功耗却更低。

图 13. 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，您还可以降低峰值电流。

有关 PSoC 时钟的更多信息，请参考 PSoC 3 和 PSoC 5LP 时钟资源应用笔记。

3.2 32 kHz 晶体的低功耗模式

当 PSoC 器件处于睡眠模式时，可以配置 32 KHz 晶振工作在低功耗模式，它默认工作在正常功耗模式。在 main() 函数的入口附近调用如下函数可实现此操作：

```
CyXTAL_32KHZ_SetPowerMode(1);
```

在活动模式和备用活动模式期间，该晶体继续以正常功耗进行操作。在休眠模式下，它被禁用。在低功耗模式下的电流比正常模式下的电流小~1 uA。

更多有关该函数的信息，请参阅系统参考指南。

3.3 PSoC 睡眠模式下的低压检测（LVD）中断

可使用低压中断（LVI）子系统将 PSoC 器件从睡眠模式唤醒。使能该子系统需要消耗 ~1uA 左右的电流。如果您的应用程序在活动模式下使用了 LVI，但在睡眠模式下不需要该系统，那么可以通过禁用它来节省功耗。使用以下 API 函数来禁用此功能：

```
CyVdLvDigitDisable(); /* Digital LVI */
CyVdLvAnalogDisable(); /* Analog LVI */
```

器件被唤醒后，可以使用下面的函数重新使能该功能：

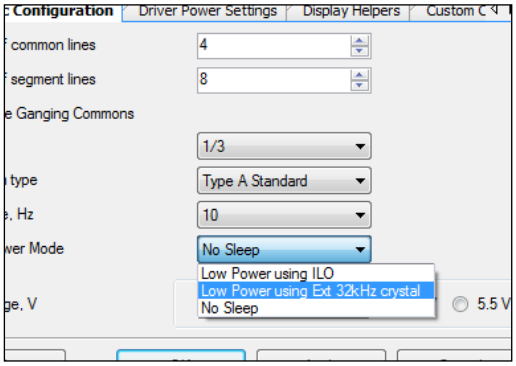
```
CyVdLvDigitEnable(<Reset>,<Threshold>);
CyVdLvAnalogEnable(<Reset>,<Threshold>);
```

默认情况下，LVI 子系统被禁用。更多有关这些函数的信息，请参阅系统参考指南。

3.4 PSoC 睡眠模式下的 SegLCD 驱动器

直接驱动段式 LCD（SegLCD）组件具有一个低功耗配置，通过它能够确保 PSoC 器件处于睡眠模式时，LCD 被刷新，如图 14 所示。

图 14. SegLCD 配置向导



不能使用 CyPmSaveClocks()函数，并且必须配置项目以便使 SegLCD 的工作频率为 12 MHz，使之能够在睡眠模式下正常操作。这是因为 CyPmSaveClocks()函数关闭了该组件所使用的数字时钟，CyPmSleep()函数会设置系统时钟，以便获得由 IMO 提供的 12 MHz 时钟。

PSoC Creator 包含两个 SegLCD 示例项目，用于说明 SegLCD 在活动模式和睡眠模式下的操作。请参考这些项目，从而深入了解如何在一个系统中实现处于睡眠模式的 SegLCD 组件。更多有关信息，请参考 AN52927 — PSoC 3 和 PSoC 5LP Segment LCD 直接驱动。

3.5 PSoC 3 升压转换器的低低功耗模式

PSoC 3 处于任何功耗模式时都可以使用 PSoC 3 升压转换器。它有两种操作模式：

- PSoC 3 处于活动模式或备用活动模式时，该升压转换器将使用升压活动模式。在该模式下，升压转换器监测着输出电压并支持所有 PSoC 功能。
- 升压待机模式是一个低功耗状态，在该模式下，升压转换器会为 PSoC 器件提供足以在睡眠模式下运行的电流。

不建议在休眠模式下使用升压转换器，因为休眠模式下的芯片功耗极低，升压器本身的功耗远远大于这个值。

当升压转换器处于升压待机模式时，PSoC 器件在活动或备用活动模式下的工作时间不能超过几微秒（取决于配置）。在 PSoC 器件进入睡眠之前，应立即将升压转化器从升压模式切换到待机模式。 您还应该在芯片唤醒后尽快将升压转换器恢复至升压模式。

```
/* Set system clocks for low power. */
CyPmSaveClocks();
/* Set boost to Standby mode. */
BoostConv_SetMode(BoostConv_BOOSTMODE_STANDBY);
/* Sleep PSoC 3 until wakeup event. */
CyPmSleep(PM_SLEEP_TIME_NONE, PM_SLEEP_SRC_BOOSTCONVERTER);
/* Restore boost to Active mode. */
BoostConv_SetMode(BoostConv_BOOSTMODE_ACTIVE);
/* Restore system clocks. */
CyPmRestoreClocks();
```

外部 32 kHz 晶体用于为转换器在睡眠模式下执行自动刷新（或“thump”）操作提供时钟。如果使用 API 函数使升压转换器进入待机模式，那么“thump”将被默认使能。

PSoC 3 数据手册与 TRM 提供了升压子系统的其他信息。有关升压转换器组件的更多信息，请参考升压转换器组件数据手册和系统参考指南。

3.6 PSoC 5 LP 升压转换器的低功耗模式

PSoC 5LP 处于任何功耗模式时都可以使用 PSoC 5LP 升压转换器。它有两种操作模式：

- 如果该芯片处于活动、备用活动或睡眠模式，那么升压转换器将进入升压活动模式。在该模式下，升压转换器监测着输出电压并支持所有 PSoC 功能。
- 升压待机模式是一个低功耗状态，在该模式下，升压转换器会为 PSoC 器件提供足以在睡眠模式下运行的电源。

不建议在休眠模式下使用升压转换器，因为休眠模式下的芯片功耗极低，升压器本身的功耗远远大于这个值。

当升压转换器处于升压待机模式时，PSoC 器件在活动或备用活动模式下的工作时间不能超过几微秒（取决于配置）。在 PSoC 器件进入睡眠之前，应立即将升压转化器从升压模式切换到待机模式。

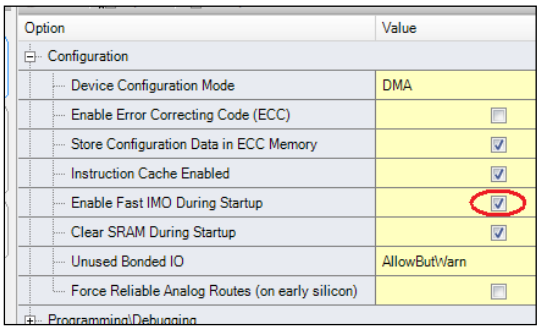
```
/* Set system clocks for low power. */
CyPmSaveClocks();
/* Set boost to Standby mode. */
BoostConv_SetMode(BoostConv_BOOSTMODE_SLEEP);
/* Sleep until boost refresh is needed. */
CyPmSleep(PM_SLEEP_TIME_NONE, PM_SLEEP_SRC_CTW);
/* Restore boost to Active mode. */
BoostConv_SetMode(BoostConv_BOOSTMODE_ACTIVE);
/* Restore system clocks. */
CyPmRestoreClocks();
```

PSoC 5LP 数据手册与 TRM 提供了升压子系统的其他信息。有关升压转换器组件的更多信息，请参考升压转换器组件数据手册和系统参考指南。

3.7 快速 IMO 启动

PSoC 3 和 PSoC 5LP 具有快速 IMO（FIMO）功能，通过该功能可以使 IMO 的启动频率为 48 MHz，从而加快器件启动的速度。因此器件启动时所消耗的电流大于正常运行时的电流。在 PSoC Creator 项目的 Design-Wide Resources（设计范围资源）文件中，默认使能了该功能，如图 15 所示。

图 15. ".cydwr"选项卡中的快速 IMO 选项



禁用该功能可降低启动时所需电流，但会延长器件的启动和初始化时间。请参考 PSoC 3 与 PSoC 5LP 架构 TRM 的"时钟系统"一节，以深入了解 FIMO 功能和它对启动时间的影响。

3.8 PSoC 睡眠模式下的看门狗

看门狗定时器可运行在活动、备用活动以及睡眠模式下。低功耗看门狗具有三个主要的配置选项：

- 无变化 — 看门狗继续以特定间隔运行，达到该间隔前必须清空看门狗。
- 最大间隔 — 看门狗继续运行并被清空，但间隔被设为最大值（1024 个计时）。唤醒器件后看门狗将首次被清空，此时间隔将被设为原始值。
- 禁用 — 在 PSoC 器件处于低功耗模式期间，看门狗将被禁用。器件被唤醒时，看门狗被使能并以特定间隔运行。

建议使用 API（CyWdtStart），以确保按需配置看门狗。"最大间隔"选项在 PM_WDT_CFG 寄存器中被默认设置。

看门狗在休眠模式下无效。从休眠模式唤醒后，它将被复位，但行为可能与用户固件中定义的配置不匹配。

更多有关看门狗定时器的操作及相关 API 的信息，请参考 PSoC 3 和 PSoC 5LP 架构 TRM 和系统参考指南中的内容。

3.9 PSoC 低功耗模式下的 GPIO

当 PSoC 器件处于低功耗模式时，GPIO 可以继续驱动。 当您需要将其他外部逻辑保持在固定电平时，这很有用，但如果引脚不必要地提供或吸收电流，则可能导致功率浪费。

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 是最佳选择，那么可以使用引脚组件的 Write 函数来设置它。

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

应配置所有未使用的 GPIO 为模拟高阻状态，除非有特殊原因需要使用其他驱动模式。如果所有与引脚组件相关的物理引脚都被配置为模拟高阻态，那么您可以使用组件的 SetDriveMode 函数。

```
/* Set My Pin to Alg Hi-Z for low power. */
MyPin_SetDriveMode(MyPin_DM_ALG_HIZ);
```

如果只需要更改某些与引脚组件相关的引脚，那么可以写入到端口引脚配置寄存器内。PSoC 器件上的每个 GPIO 引脚都有一个寄存器，名为 CYREG_PRTx_PCy，其中"x"是端口号，"y"是引脚号。

借助于 PSoC 3 和 PSoC 5LP 的灵活性，用户可方便管理 GPIO 驱动模式，避免不必要的漏电流。请参考 AN72382 — 使用 PSoC 3 和 PSoC 5LP GPIO 引脚，以获得更多信息。

3.10 PSoC 低功耗模式下的 SIO

在将 PSoC 器件置于休眠模式之前，应将特殊输入/输出（SIO）引脚置于单端模式，以降低 PSoC 休眠电流。 差分模式下的 SIO 消耗 100µA 的高电流。 要将 SIO 置于单端模式，请使用寄存器 PRT12_SIO_CFG 并将特定 SIO 对的位设置为零。 PSoC 器件退出休眠模式后，应将这些位设置回以前的值。表 2 显示了 SIO 配置寄存器的描述。

表 2. PRT12_SIO_CFG 寄存器

SIO[7:6]		SIO[5:4]		SIO[3:2]		SIO[1:0]	
位 7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	位 0

请参考 AN60580 — PSoC 3/PSoC 5LP 中的 SIO 提示与技巧，以获得更多信息。

此外，如果您将 SIO 引脚作为输出使用，并且将“驱动电平”设置为“Vref”而不是“Vddio”，您则可能需要在低功耗模式期间禁用 Vref 模式以最小化功耗。这些操作是成对完成的，因此仅在两个相邻引脚的组中启用 Vref 输出模式。请参阅下面的代码示例，它定义了引脚对的掩码以及如何启用或禁用 Vref。

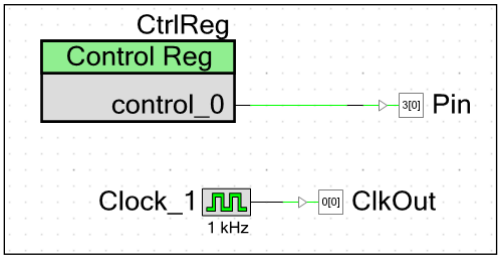
```
#define SIO_VREG_EN_1_0 0x01 // Mask for SIO pins[1:0]
#define SIO_VREG_EN_3_2 0x04 // Mask for SIO pins[3:2]
#define SIO_VREG_EN_5_4 0x10 // Mask for SIO pins[5:4]
#define SIO_VREG_EN_7_6 0x40 // Mask for SIO pins[7:6]

// Code to enable and disable Vref output mode (Instance name is “MySIO”)
MySIO_SIO_CFG |= VREG_EN_3_2; // Set to use Vref reference
MySIO_SIO_CFG &= ~VREG_EN_3_2; // Clear Vref for Low Power mode
```

3.11 PSoC 低功耗模式下的数字模块

PSoC 器件处于睡眠和休眠模式期间，某些子系统始终处于关闭状态。将这些子系统连接到保持充电（打开）状态的其他子系统可导致意外行为。

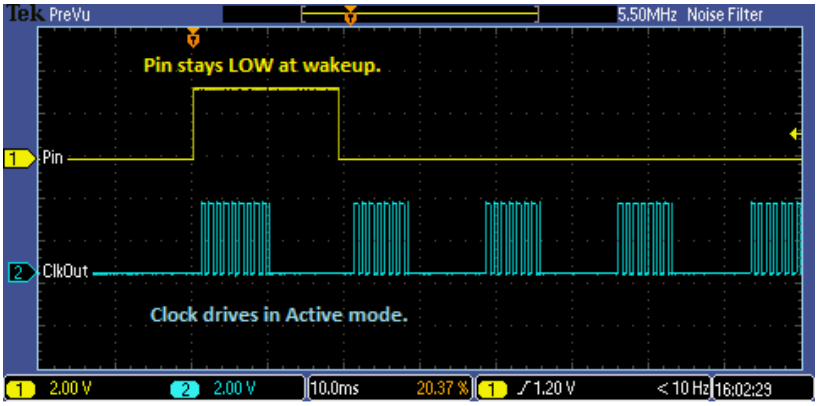
图 16 显示的是控制寄存器，该寄存器用于设置 GPIO 引脚的输出（其他引脚和时钟用于指示 PSoC 器件已被唤醒）。

图 16. 用于设置引脚输出的控制寄存器



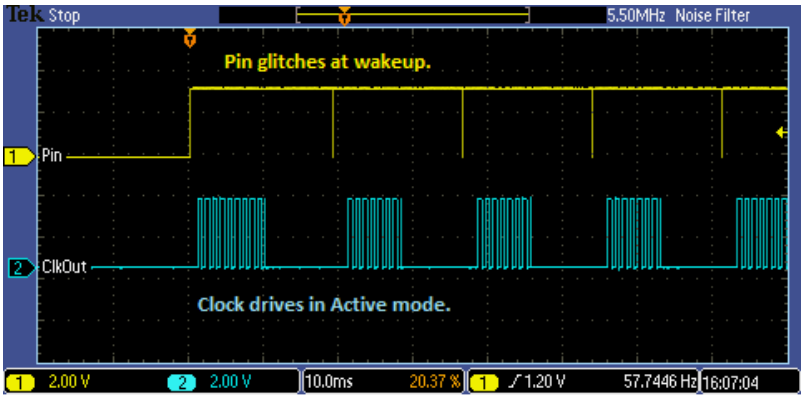
当芯片处于低功耗模式时，引脚继续驱动其最后状态（HIGH），但包含控制寄存器的物理模块未通电。 当 PSoC 器件唤醒并恢复数字模块的电源时，控制寄存器位复位为 0。 连接到控制寄存器的引脚然后从高电平切换到低电平输出，如图 17 所示。

图 17. 控制寄存器不会持续整个芯片睡眠模式过程



即使在唤醒器件时将控制寄存器位设置为'1'，仍然会发生干扰。这是由于在固件可以进行修改前，器件唤醒期间引脚逻辑在短时间内仍保持为'0'，如图 18 所示。

图 18. 每次器件被唤醒时存在的控制寄存器干扰



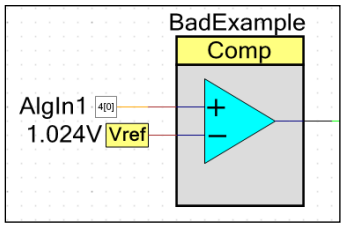
在这种情况下，应该直接控制引脚，而不是通过控制寄存器来控制它。

3.12 PSoC 低功耗模式下的参考源

在 PSoC 中，在睡眠或休眠模式期间，VDDA、VDDD、和 VBAT 保持连接状态，但其他 VREF 源则不会保持该状态。此外，VDDA/2 参考电压保持有效状态，但电压分频器则断开连接。如果在睡眠或休眠模式期间需要参考电压，必须使用一个外部源。

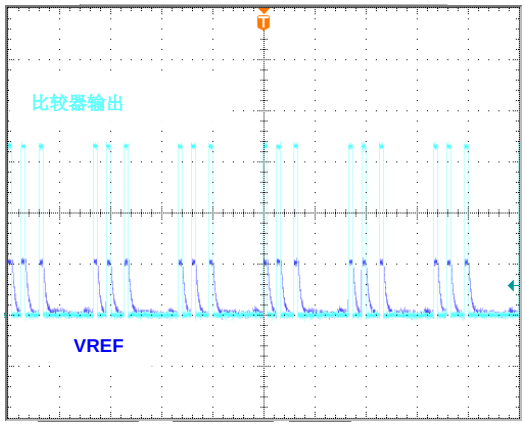
图 19 显示的是这种情况的示例。将比较器配置为唤醒源，它拥有一个连接到负端的 VREF 组件。

图 19. 如何在低功耗模式下不使用 VREF



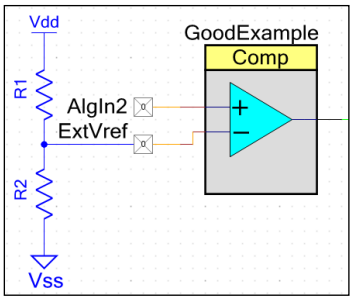
只要 PSoC 器件处于活动或备用活动模式，这种情况便可行，只是在低功耗模式下 VREF 会断开连接。因此当负端处于悬空状态时，器件会被间歇性唤醒，如图 20 所示。

图 20. 比较器输入处于悬空状态时器件被间歇性唤醒（电压比例 = 1.0 V）



通过提供外部参考电压可解决该问题，如图 21 所示。PSoC 器件处于睡眠模式期间，该参考电压保持为有效状态，因该比较器唤醒源可以按要求运行。

图 21. 在低功耗模式下使用外部 VREF



输入终端需要的电流很小，因此可以使用较大的电阻来限制通过分频器流失的电源。

3.13 睡眠和休眠模式下的电压调节器

PSoC 3 和 PSoC 5LP 有两个低功耗电压调节器，它们用于保持在睡眠和休眠模式下的逻辑状态。

- 在睡眠模式下的电压调节器为组件提供足够大的电源，从而可以进行快速唤醒并满足子系统的要求，这些子系统在深度睡眠模式下保持为活动状态。
- 在休眠模式下，休眠电压调节器所提供的电源仅满足用于保持最基础的寄存器、存储器以及锁存器的逻辑状态。

可以在 VCCD 和 VCCA 引脚上获得睡眠电压调节器的输出，但它提供的电流仅满足为内部 PSoC 资源供电。不应该使用睡眠电压调节器来进行任何其他目的。

休眠电压调节器不给 VCCD 和 VCCA 网络供电，所以无法观察到它的输出。

3.14 芯片编程时的功耗

编程期间，PSoC 3 和 PSoC 5LP 消耗的电流约为 14 mA。这是因为编程和调试逻辑与时钟和 CPU 一起启用。

如果您的设计不能提供该级别的电流，那么在编程或调试期间，必须通过外部源（如给 MiniProg3）为 PSoC 器件供电。请参考 PSoC Creator Help 文件，深入了解如何配置 MiniProg3，以为 PSoC 供电。

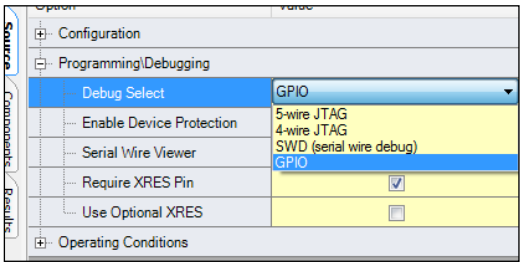
3.15 调试接口是否使能

PSoC 3 和 PSoC 5LP 支持片上调试。在调试模式下，实际消耗的电流可能会大于您的预期值。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 或者 JTAG 模式，并且连接了 MiniProg3，即使 PSoC 器件并不处于调试模式，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 22 所示。

图 22. 禁用调试接口从而降低功耗



即使引脚被设置为 GPIO 模式，您仍可以对它们进行编程和调试。如果调试接口被禁用，必须经过复位后才能访问 PSoC 器件内的调试控制器。因此不能将正在执行的项目连接到调试器上。

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，请查看器件数据手册和 TRM 内容。

4 近似功耗

器件数据手册和组件数据手册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，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。可以在赛普拉斯网页上找到该电子表格（*PSoC3_5_Power_Estimator.xls*），该网页还提供了本应用笔记 — AN77900。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，您已经仔细阅读了 **Instructions** 选项卡。

5 总结

适当的低功耗技术会对便携式设备产生重大影响。它可以降低器件电池的大小，从而减小产品的大小，或者延长特定电池的寿命。通过使用在 PSoC 3 和 PSoC 5LP 中的多个节省功耗的技巧，您可以优化设计，并确保它消耗的功耗最低。

6 相关应用笔记

这些应用笔记提供了有关本文档中未完整说明的主题的更多信息：

- AN86233 — PSoC 4 低功耗模式和降低功耗技术
- AN54181 — PSoC 3 入门
- AN77759 — PSoC 5LP 入门
- AN77835 — 从 PSoC 3 升级到 PSoC 5LP 的指南
- AN61290 — PSoC 3 和 PSoC 5LP 硬件设计中的注意事项
- AN54460 — PSoC 3 和 PSoC 5LP 的中断
- AN60616 — PSoC 3 和 PSoC 5LP 的启动过程
- AN60631 — PSoC 3 和 PSoC 5LP 的时钟资源
- AN72382 — 使用 PSoC 3 和 PSoC 5LP GPIO 引脚
- AN60580 — PSoC 3/PSoC 5LP 中 SIO 的提示和技巧
- AN52705 — PSoC 3 和 PSoC 5LP — DMA 入门
- AN52927 — PSoC 3 和 PSoC 5LP — 段式 LCD 直接驱动

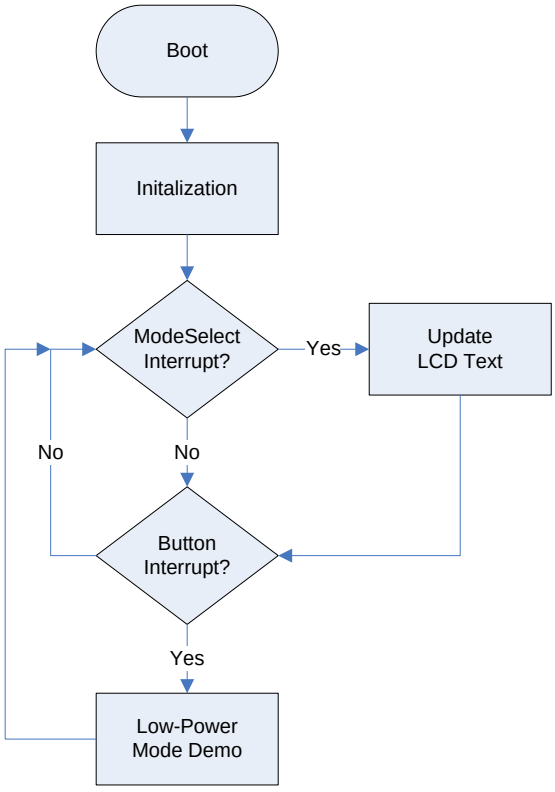
附录A. 低功耗示例项目

使用 PSoC 是熟悉它的最佳选择之一。四个示例项目都与本应用笔记相关。它们是同一个 PSoC Creator 工作区中的所有组成部分，可以运行于同一个开发套件（DVK）硬件设置上。

A.1 两个低功耗演示项目

这些项目展示了 PSoC 3 和 PSoC 5LP 功耗模式转换和唤醒源。通过将 Port 0 引脚设置为一个特定值来选择功耗模式和唤醒源。在 P2[7]的下降沿上，PSoC 器件能够执行代码，从而演示所选定的配置，如图 23 所示。

图 23. PSoC 低功耗演示流程图



这些项目演示了所有通用睡眠和休眠唤醒源。SegLCD 在 PSoC Creator 中拥有自己的示例项目，并且在本应用笔记的主要内容中详细说明了睡眠模式期间的升压调节。只展示了两个备用活动源，因为在睡眠模式下它们基本相同。表 3 列出了这些示例项目所演示的功耗模式和唤醒源。

表 3. PSoC 演示唤醒源

模式	唤醒源
活动模式	无
备用活动模式	PICU
	RTC
睡眠模式	PICU
	RTC（使用 OPPS）
	SleepTimer（睡眠定时器 — 使用 CTW）
	CTW（仅适用于 PSoC 3）
	比较器
	I^2C 地址
	LCD（未实现）
休眠模式	PICU
自定义	无（由用户定义）

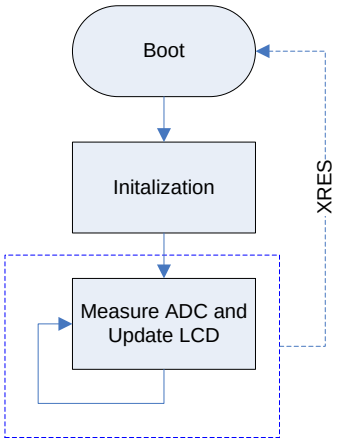
该代码演示了本应用笔记中介绍的原则和技巧。尚未尝试优化该代码的大小和速度。在该项目的原理图和源文件内提供了项目使用指导内容。

A.2 电压警报 — 无优化

该项目使用 PSoC 3 器件展示了简单的电压测量和警报系统，如图 24 所示。尚未针对低功耗优化该项目。默认时钟和全局功耗设置保持不变，并未使用任何低功耗模式。

DelSig ADC 读取模拟输入端的值，并且读取结果将显示在 LCD 上。RTC 组件也进行操作，并且时间值被显示在 LCD 上。如果输入上的电压超过已定义的等级，则 LED 将开始闪烁发光，并且“Alarm”（警报）将显示在 LCD 上。

图 24. 警报系统 — 未优化

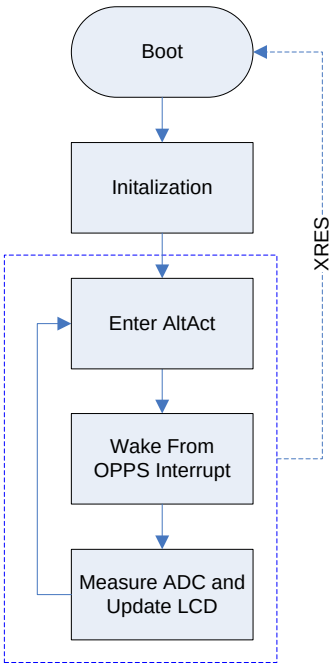


该项目的原理图和源文件内提供了项目使用指导内容。

A.3 电压警报 — 优化

该项目演示了一个示例，该示例使用与上述的电压测量和警报示例相同的基本功能。实现了一些低功耗优化以降低功耗，如图 25 所示。

图 25. 警报系统 — 优化



该项目的原理图和源文件内提供了项目使用指导内容。

附录B. DVK 上的电源测量

赛普拉斯开发套件（DVK）演示了 PSoC 器件提供的模拟和数字功能。DVK 未经优化，因此不便测量 PSoC 器件的电流消耗。本附录介绍了如何修改 CY8CKIT-001、CY8CKIT-030 和 CY8CKIT-050 电路板，从而正确测量 PSoC 电源。进行修改前，请确保您已经熟悉了这些电路板的原理图。在单独套件的网页上提供了各种原理图。

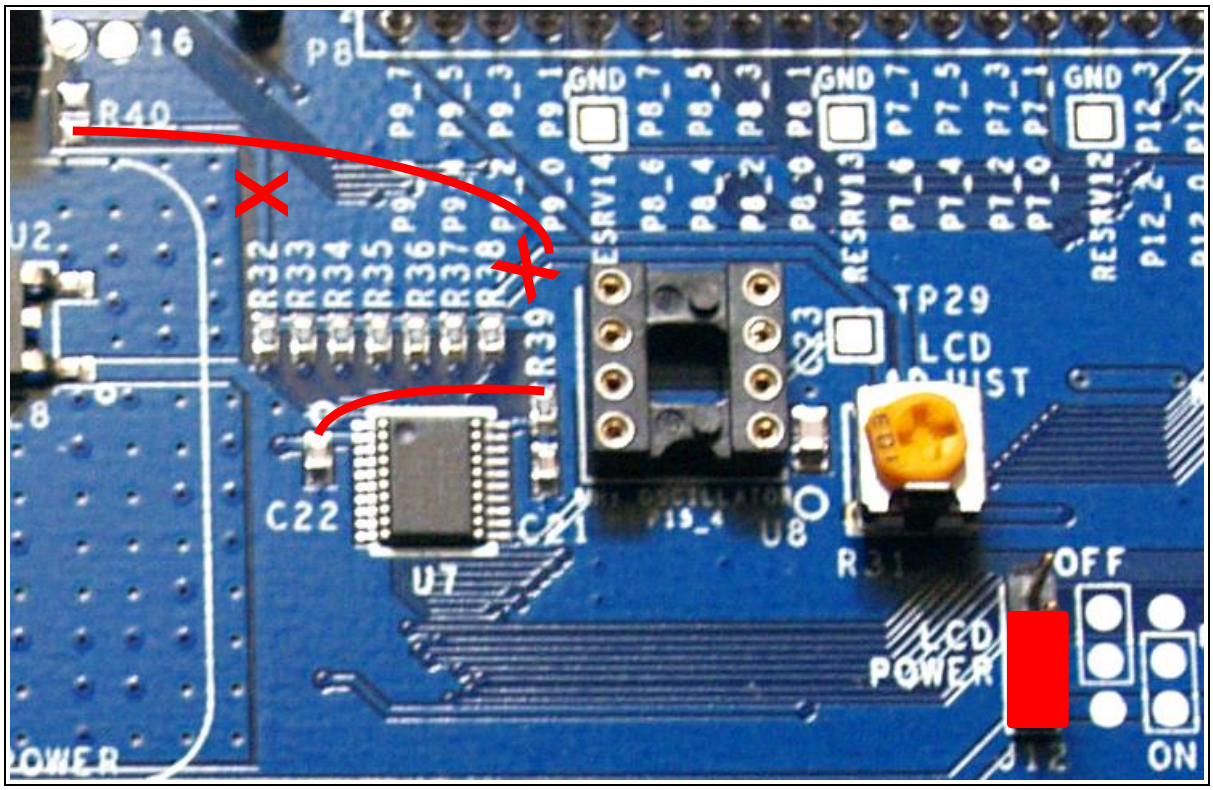
B.1 对 CY8CKIT-001 进行修改

下列步骤演示了如何修改 CY8CKIT-001 开发板从而正确测量电流。这里介绍的配置将单个电源轨应用给 VDDD、VDDA 和 VDDIO。通过将仪表串连到 PSoC VDDD 引脚可以测量电源。

1. 移除 J2、J3、J4、J5、J6、J7、J10 和 J11 上的跳线器。
2. 将 J6 的中心引脚与 J7 的中心引脚短接，从而使 VDD_DIG 与 VDD_ANLG 网络相连。
3. 将 J2、J3、J4 和 J5 的中心引脚与 J6 的中心引脚短接，从而使 VDDIO 网络与 VDD_DIG 相连。
4. 将 J11 的引脚 1（“VR”字上面）与 J10（“RS232”字旁边）的引脚 1 短接。这样，VDD 网络（而不是 VDD_ANLG）将给电位器 R20 供电。
5. 将万用表的低电平端连接到 J7 的中心引脚。
6. 将万用表的高电平端连接到 VDD 监测点。

   注意： 如果您的工作电压为 5 V，那么无需进行其他修改。如果工作电压为 5 V，LCD 电平转换器电路可以给 PSoC I/O 引脚供电并降低在睡眠或休眠模式期间的功耗。下列步骤可防止发生这种情况，并允许字符 LCD 在 PSoC 器件的工作电压为 3.3 V 或 5 V 时正常工作。预先警告，进行该修改时需要切断 PCB 上的走线。第 24 页上的图 26 显示了下列步骤中介绍的连接端。
7. 切断经过开发板顶层上文字“R32”的走线。在走线转向 R40（附近没有任何过孔或组件）前，建议切断经过“R32”文字的走线。
8. 再次切断经过文字“R38”和文字“RESRV14”间的同一条走线。切断这两条导线会使高侧移位器信号上的上拉电阻断开与 VCC_LCD 网络的连接。
9. 在离文字“RESRV14”最近的一侧，刮掉切断导线处的阻焊膜，露出铜箔用于焊接。
10. 将 R40 底盘（“R40”文字下方）中的跳线焊接到前一步骤刮掉阻焊膜的铜箔上。该操作会绕过前两次切断操作所断开的部分，并将 LCD 模块的电源重新连接至 J12 的中心引脚上。
11. 将 R39 焊针（“R39”文字下方）中的跳线焊接到 C22 的焊盘上（U7 的圆孔丝印旁边）。这样会将高侧移位器的上拉电阻和参考引脚连接 3.3 V 的电压。
12. 将 J12 上的跳线器设置为 ON，以使能 LCD 模块。

图 26. 旁路 CY8CKIT-001 LCD3.3 V 电平转换器，以便在工作电压为 3.3 V 时进行低功耗测量



完成这些修改后，可以准确测量 PSoC 器件所使用的总电流。KIT-001 电路板上的所有其他组件均由 VDDD 单独供电，因此无法观察它们的功耗。通过使用 SW3，将 CY8CKIT-001 配置为 5 V 或 3.3 V 即可执行正常的操作（不需要进行其他更改）。

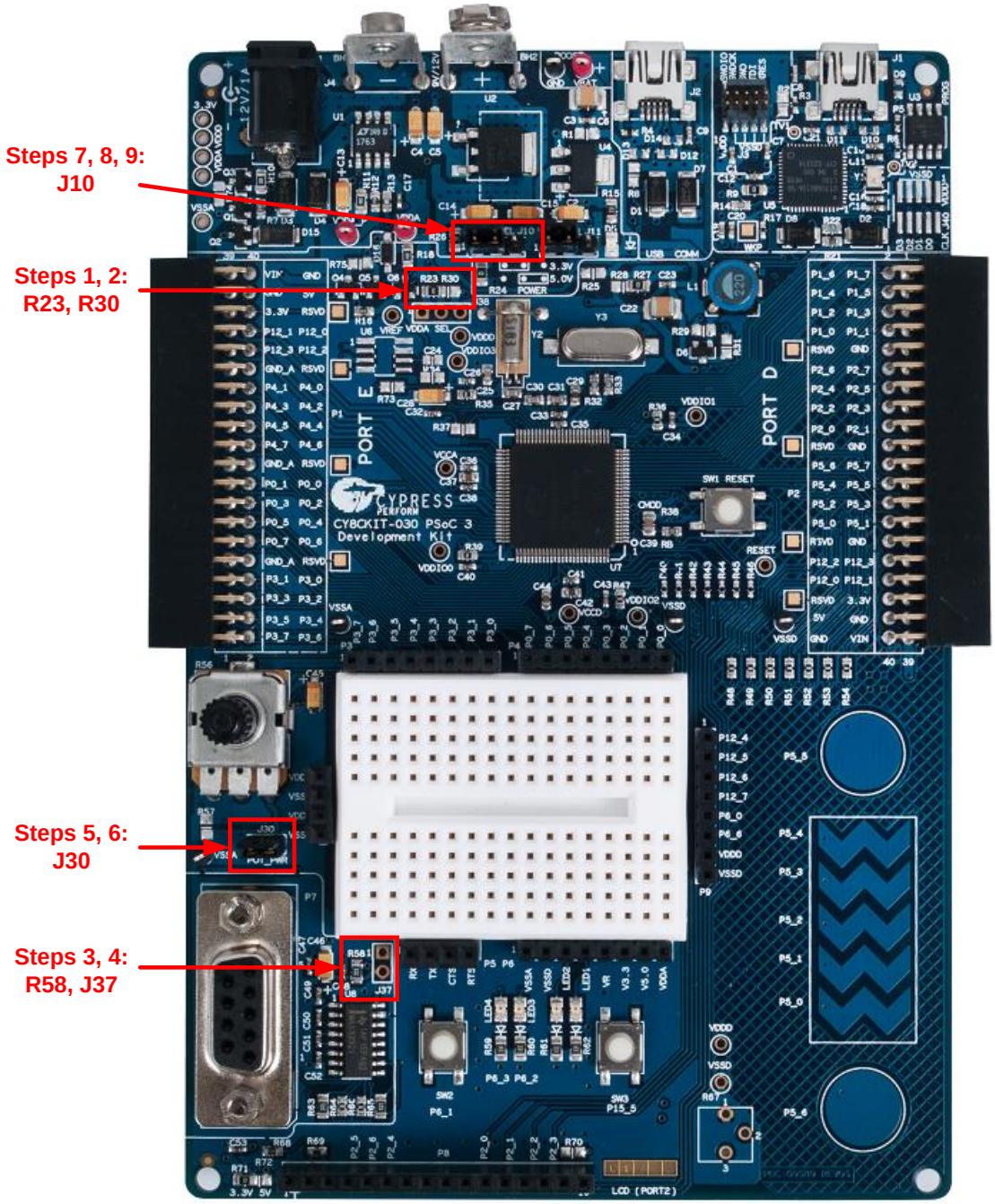
B.2 CY8CKIT-030 和 CY8CKIT-050 的修改

下列步骤演示了如何修改 CY8CKIT-030 和 CY8CKIT-050 开发板以正确测量电流。第 25 页上的图 27 显示的是“Called-out”硬件。这里介绍的配置会将单个电源轨用于 VDDD、VDDA 和 VDDIO。通过将仪表串连到 PSoC VDDD 引脚上可以测量电源。

1. 移除 R23，从而使 VDDA 网络与 VDDA_P 网络断开连接。
2. 在 R30 的焊盘上焊接一个零欧姆电阻，从而使 VDDD 与 VDDA 相连。这样您可以测量 PSoC 器件所消耗的总电流。
3. 清除 R58，从而使 RS-232 驱动器 U8 与 VDDA 断开连接。默认情况下，U8 由 VDDA 供电，因此需要测量它的电压。
4. 如果需要 RS-232，请将 J37 的短路引脚 2（“J37”文字上面）连接到 VDDA_P 测试点。这样，RS-232 驱动器将由 VDDA_P（而不是 VDDA）供电，因此不需要计算它的电压。
5. 移除 J30 上的跳线器。默认情况下，R56 由 VDDA 供电，因此不需要计算它的电压。
6. 如果需要电位器 R56，那么将 J30 的短路引脚 2（“J30”文字下面）连接到 VDDA_P 测试点上。这样，该电位器将由 VDDA_P 供电。
7. 移除 J10 上的跳线器。
8. 将万用表的低电平端连接到 J10 的中心引脚。
9. 将万用表的高电平端连接到 J10 的一个外侧引脚（用以选择 5 V 或 3.3 V 操作）。

完成这些修改后，可以准确测量 PSoC 器件所使用的总电流。开发板上的所有其他组件均由 VDDD 单独供电，因此不需要测量它们的电压。使用 J10 和 J11 将 DVK 配置为 5 V 或 3.3 V 即可正常进行操作（不需要进行其他更改）。

图 27. CY8CKIT-030/050 低功耗修改



附录C. 电源管理 API 和寄存器

赛普拉斯提供了各种 API 子程序，用于使 PSoC 器件在各个低功耗模式之间进行切换。这里介绍的都是最常用的函数。它们的格式为 *CyPm.c*，它是每个 PSoC Creator 项目的一部分。

系统参考指南详细说明了电源管理 API，而 PSoC 3 和 PSoC 5LP 技术参考手册详细介绍了各种寄存器。

C.1 CyPmSaveClocks()

该函数为低功耗操作提供 PSoC 时钟。进入睡眠或休眠模式前，应该立即调用该函数。而进入备用活动模式前，通常不会调用该函数。这是因为各种时钟需要保持正常的运行状态。

如果在进入睡眠或休眠模式前无法调用该函数，将导致未定义的结果。

C.2 CyPmRestoreClocks()

通过调用该函数可以使 PSoC 器件恢复调用 CyPmSaveClocks()时的设置。它通常是退出低功耗模式后第一个被调用的函数。

C.3 CyPmAltAct()

该函数用于管理备用活动模式下的转换过程。它拥有两个参数：wakeupTime 和 wakeupSource。参数 wakeupTime 用于设置定时器的频率，并取消屏蔽将其作为唤醒源。参数 wakeupSource 用于取消屏蔽异步和基于组件的唤醒源。

注意： 如果将“中断”配置为备用活动模式唤醒源，则您不能屏蔽单独中断组件。另外，在中断控制器滤波原始中断前，电源管理器将观察这些中断。这便意味着通常应用于中断信号的边沿检测或使能设置都会被忽略。

C.4 CyPmSleep()

该函数用于管理睡眠模式的转换过程。它有两个参数：wakeupTime 和 wakeupSource。参数 wakeupTime 用于设置定时器的频率，并取消屏蔽将其作为唤醒源。参数 wakeupSource 用于取消屏蔽异步和基于组件的唤醒源。

在调用该函数之前，您应该调用 CyPmSaveClocks()函数。这是为了确保在进入睡眠模式之前准确保存了时钟状态。

C.5 CyPmHibernate()或 CyPmHibernateEx()

CyPmHibernate()用于管理休眠模式的转换过程。它不带有任何参数，使能 PICU 和 XRES 为有效唤醒源。

CyPmHibernateEx()有一个参数用于从休眠模式更改为唤醒源。它们可以配置为 PICU 中断、Comparator0、Comparator1、Comparator2 和 Comparator3 输出。这允许用户指定确切的唤醒源，并且在不同的操作模式要求时，则可以容易地改变它们。该函数基本上允许设计掩盖不需要或不要求的唤醒源。

在调用这些函数前，您应该调用 CyPmSaveClocks()函数。这是为了确保在进入休眠模式前准确保存了时钟状态。另外，从休眠模式唤醒后，至少要经过 20 µs 时间 PSoC 才能重新进入低功耗模式，以确保睡眠调节器处于稳定状态。

C.6 组件低功耗 API

几乎所有 PSoC Creator 组件都有 API 函数，用于将组件进入低功耗模式。Sleep 函数保存了组件设置，然后调用 Stop 函数。调用 Stop 和 Sleep 函数的节能情况不存在差异。

除非该组件作为唤醒源使用，否则在调用 CyPmSleep()或 CyPmHibernate()函数前应该调用它的 Sleep 函数。Sleep 函数的格式为：

```
MyComponentName_Sleep();
```

拥有 Sleep 函数的所有组件均有一个 Wakeup 函数，用于恢复组件之前的状态。其格式为：

```
MyComponentName_Wakeup();
```

Sleep 和 Stop 函数不会检查组件的闲置或繁忙状态。您应该检查代码，以确保有效组件不繁忙。更多有关如何管理低功耗操作的信息，请参考单独组件数据手册中介绍的内容。

C.7 寄存器的直接写操作

应尽可能使用 API 函数来控制功耗模式的切换过程，但可以使用寄存器写操作来代替。在这里提到的寄存器是与功耗模式切换相关的最常用的寄存器。

C.7.1 PM_MODE_CSR

功耗模式控制和状态寄存器是与 PSoC 功耗模式相关的最重要寄存器。该寄存器中的位[2:0]控制着整个 PSoC 器件的全局功耗，如 27 页上的表 4 所示。通过读取这些位，您可以知道 PSoC 器件当前所处的模式。

表 4. PM_MODE_CSR[2:0]值

[2:0]值	模式
000	活动模式
001	备用活动模式
010	（不支持）
011	睡眠模式
100	休眠模式
101	（不支持）
110	（不支持）
111	（不支持）

设置这些位不会自动将整个 PSoC 器件配置为低功耗操作模式。更多有关全局功耗设置以及这些设置是如何影响 PSoC 3 和 PSoC 5LP 子系统的信息，请参考 PSoC 3 和 PSoC 5LP 技术参考手册中介绍的内容。

注意： 使用不支持的低功耗模式或唤醒源会导致不可靠的结果。应该尽可能使用标准的 API 进入低功耗模式。

C.7.2 PM_TW_CFGx

这三个寄存器可使能和配置定时器（通过这些定时器可以将 PSoC 从低功耗模式唤醒）。

C.7.3 PM_WAKEUP_CFGx

这三个寄存器用于屏蔽功耗模式唤醒源。通过动态屏蔽或取消屏蔽单独源可以在不同的条件下提供不同的唤醒源。

C.7.4 PICUx_INTTYPEy

这些寄存器用于配置 PICU。每个引脚可以使用一个寄存器来设置中断触发条件。

C.7.5 PICUx_INTSTAT

这些寄存器保持引脚中断的状态。每个端口存在一个状态寄存器，该寄存器的每一位与一个引脚相应。

C.7.6 FASTCLK

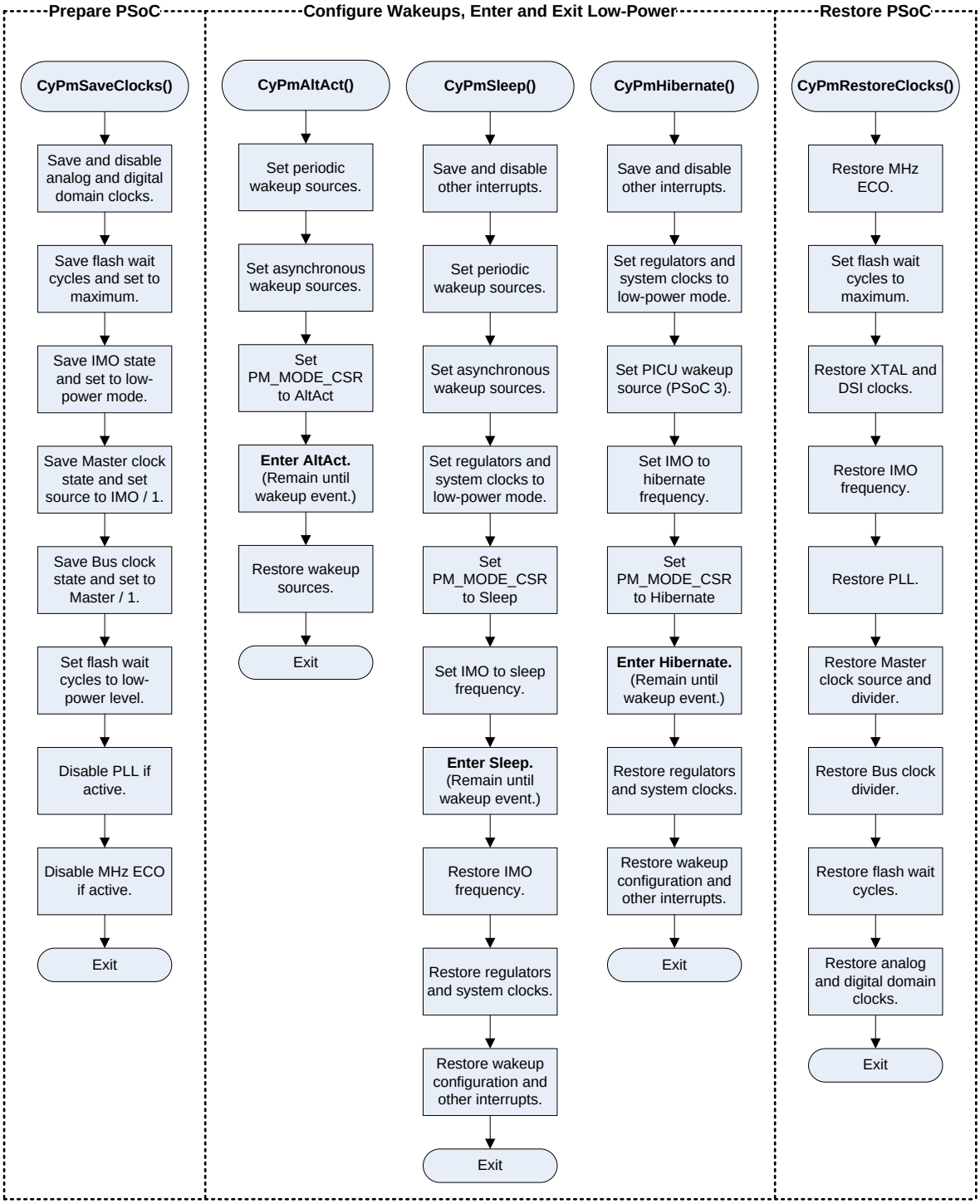
这些寄存器用于配置 IMO、主设备时钟和 PLL。时钟分配是由 CLKDIST 寄存器控制的。

C.7.7 SLOWCLK

这些寄存器用于配置内部低速振荡器（ILO）和 32 kHz 晶体。时钟分配是由 CLKDIST 寄存器控制的。

C.8 电源管理 API 流程图

图 28. 电源管理函数流程图



C.9 电源管理 API 寄存器参考

表 5 列出了 PSoC 3 和 PSoC 5LP 器件中所提供的异步唤醒源。

表 5. CyPm 唤醒源配置

API	唤醒源	定义的 wakeupSource 屏蔽	受影响的寄存器[位]
CyPmAltAct()	比较器 0	PM_ALT_ACT_SRC_COMPARATOR0	PM_WAKEUP_CFG1 [0x01]
	比较器 0	PM_ALT_ACT_SRC_COMPARATOR1	PM_WAKEUP_CFG1 [0x02]
	比较器 0	PM_ALT_ACT_SRC_COMPARATOR2	PM_WAKEUP_CFG1 [0x04]
	比较器 0	PM_ALT_ACT_SRC_COMPARATOR3	PM_WAKEUP_CFG1 [0x08]
	所有中断	PM_ALT_ACT_SRC_INTERRUPT	PM_WAKEUP_CFG0 [0x01]
	PICU	PM_ALT_ACT_SRC_PICU	PM_WAKEUP_CFG0 [0x04]
	I^2C 地址	PM_ALT_ACT_SRC_I2C	PM_WAKEUP_CFG0 [0x08]
	升压	PM_ALT_ACT_SRC_BOOSTCONVERTER	PM_WAKEUP_CFG0 [0x20]
	FTW	PM_ALT_ACT_SRC_FTW	PM_WAKEUP_CFG0 [0x40]
	CTW	PM_ALT_ACT_SRC_CTW	PM_WAKEUP_CFG0 [0x80]
	OPPS	PM_ALT_ACT_SRC_ONE_PPS	PM_WAKEUP_CFG0 [0x80]
	LCD	PM_ALT_ACT_SRC_LCD	PM_WAKEUP_CFG2 [0x01]
CyPmSleep()	比较器 0	PM_SLEEP_SRC_COMPARATOR0	PM_WAKEUP_CFG1 [0x01]
	比较器 1	PM_SLEEP_SRC_COMPARATOR1	PM_WAKEUP_CFG1 [0x02]
	比较器 2	PM_SLEEP_SRC_COMPARATOR2	PM_WAKEUP_CFG1 [0x04]
	比较器 3	PM_SLEEP_SRC_COMPARATOR3	PM_WAKEUP_CFG1 [0x08]
	PICU	PM_SLEEP_SRC_PICU	PM_WAKEUP_CFG0 [0x04]
	I^2C 地址	PM_SLEEP_SRC_I2C	PM_WAKEUP_CFG0 [0x08]
	升压转换器	PM_SLEEP_SRC_BOOSTCONVERTER	PM_WAKEUP_CFG0 [0x20]
	CTW	PM_SLEEP_SRC_CTW	PM_WAKEUP_CFG0 [0x80]
	OPPS	PM_SLEEP_SRC_ONE_PPS	PM_WAKEUP_CFG0 [0x80]
	LCD 玻璃驱动	PM_SLEEP_SRC_LCD	PM_WAKEUP_CFG2 [0x01]
CyPmHibernate()	PICU	N/A	PM_WAKEUP_CFG0 [0x04]

表 6 列出了 PSoC 3 器件中所提供的各种定期唤醒源。

表 6. CyPm 唤醒周期配置

API	唤醒源	定义的 wakeupTime 屏蔽	受影响的寄存器[位]
CyPmAltAct() and CyPmSleep()1	OPPS	PM_ALT_ACT_TIME_ONE_PPS PM_SLEEP_TIME_ONE_PPS	PM_TW_CFG2 [0x20] — 取消屏蔽 PM_TW_CFG2 [0x10] — 使能
	CTW 2 毫秒	PM_ALT_ACT_TIME_CTW_2MS PM_SLEEP_TIME_CTW_2MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x01] — 周期为 2 毫秒
	CTW 4 毫秒	PM_ALT_ACT_TIME_CTW_4MS PM_SLEEP_TIME_CTW_4MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x02] — 周期为 4 毫秒
	CTW 8 毫秒	PM_ALT_ACT_TIME_CTW_8MS PM_SLEEP_TIME_CTW_8MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x03] — 周期为 8 毫秒
	CTW 16 毫秒	PM_ALT_ACT_TIME_CTW_16MS PM_SLEEP_TIME_CTW_16MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x04] – 周期为 16 毫秒
	CTW 32 毫秒	PM_ALT_ACT_TIME_CTW_32MS PM_SLEEP_TIME_CTW_32MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x05] — 周期为 32 毫秒
	CTW 64 毫秒	PM_ALT_ACT_TIME_CTW_64MS PM_SLEEP_TIME_CTW_64MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x06] — 周期为 64 毫秒
	CTW 128 毫秒	PM_ALT_ACT_TIME_CTW_128MS PM_SLEEP_TIME_CTW_128MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x07] — 周期为 128 毫秒
	CTW 256 毫秒	PM_ALT_ACT_TIME_CTW_256MS PM_SLEEP_TIME_CTW_256MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x08] — 周期为 256 毫秒
	CTW 512 毫秒	PM_ALT_ACT_TIME_CTW_512MS PM_SLEEP_TIME_CTW_512MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x09] — 周期为 512 毫秒
	CTW 1024 毫秒	PM_ALT_ACT_TIME_CTW_1024MS PM_SLEEP_TIME_CTW_1024MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x0A] — 周期为 1024 毫秒
	CTW 2048 毫秒	PM_ALT_ACT_TIME_CTW_2048MS PM_SLEEP_TIME_CTW_2048MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x0B] — 周期为 2048 毫秒
	CTW 4096 毫秒	PM_ALT_ACT_TIME_CTW_4096MS PM_SLEEP_TIME_CTW_4096MS	PM_TW_CFG2 [0x08] — 取消屏蔽 PM_TW_CFG2 [0x04] — 使能 PM_TW_CFG1 [0x0C] — 周期为 4096 毫秒
	FTW 10 µs 至 2.56 毫秒（仅在备用活动模式）	PM_ALT_ACT_TIME_FTW(1-256)	PM_TW_CFG2 [0x02] — 取消屏蔽 PM_TW_CFG2 [0x01] — 使能 PM_TW_CFG0 [0x00 到 0xFF]周期
CyPmHibernate()	N/A	N/A	N/A

[1] PSoC 5LP 仅支持 CTW 和 OPPS 唤醒源作为 SleepTimer 或 RTC 组件的部分。

文档修订记录

文档标题：AN77900 — PSoC® 3 和 PSoC 5LP 低功耗模式和降低功耗技术

文档编号：001-97875

版本	ECN	变更者	提交日期	变更说明
**	4802432	SNYQ	06/23/2015	本文档版本号为 Rev**，译自英文版 001-77900 Rev*C。
*A	6265108	XZNG	07/30/2018	本文档版本号为 Rev*A，译自英文版 001-77900 Rev*G。

全球销售和设计支持

赛普拉斯公司拥有一个由办事处、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
时钟与缓冲区	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 | s 视频 s | 博客 | 培训 | 组件

技术支持

cypress.com/support

此处引用的所有其它商标或注册商标都归其各自所有者所有。

赛普拉斯半导体公司
198 Champion Court
San Jose，CA 95134-1709

© 赛普拉斯半导体公司，2012-2018 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。没有任何电子设备是绝对安全的。因此，尽管赛普拉斯在其硬件和软件产品中采取了必要的安全措施，但是赛普拉斯并不承担任何由于使用赛普拉斯产品而引起的安全问题及安全漏洞的责任，例如未经授权的访问或使用赛普拉斯产品。此外，本材料中所介绍的赛普拉斯产品有可能存在设计缺陷或设计错误，从而导致产品的性能与公布的规 格不一致。（如果发现此类问题，赛普拉斯会提供勘误表）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。