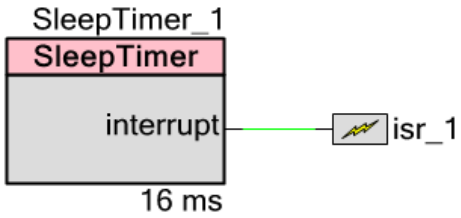
睡眠定时器

3.20

特性



- 从低功率模式唤醒器件：交替活动和睡眠
- 包含发出中断指令的可配置选项
- 在器件处于活动模式时生成定期中断指令
- 支持 12 个独立间隔：2、4、8、16、32、64、128、256、512、1024、2048 以及 4096 ms

概述

睡眠定时器元件可用一个可配置的时间间隔将器件从交替活动模式和睡眠模式唤醒。还可将它配置为以配置的时间间隔发送中断。

何时使用睡眠定时器

您可以使用睡眠定时器组件定期将器件从交替活动模式和睡眠低功耗模式唤醒。唤醒间隔时间可配置，并可以发送或不发送（仅用于 PSoC3）中断。器件处于活动模式时，您还可以将该组件作为一个计数器使用，使其定期生成中断。

硬件计数器可以执行周期中断。但这样使用硬件资源会使它的效率较低，并要求器件持续处于活动状态。

睡眠定时器则使用唯一的资源组，因此每个设计中只有一个组件可用。

中断 — 输出

睡眠定时器有一个输出连接、中断，但没有输入连接。中断输出使用中央时轮（CTW）中断源。当 CTW 计数器计数结束时，将发送一个中断。该最终计数值由模块设定或 API 函数指定。

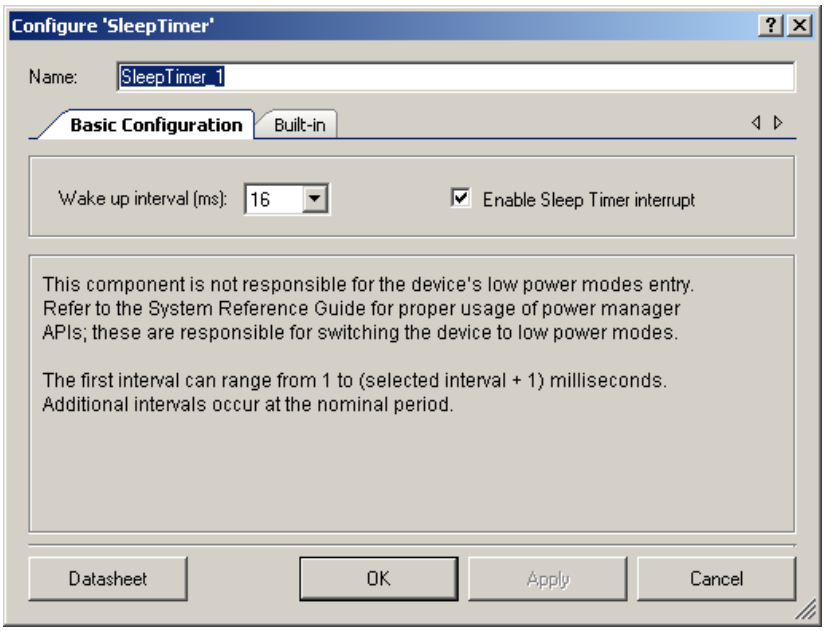
通过取消选择 **Enable Sleep Timer Interrupt** 参数，可以隐藏符号上的输出。

原理图宏信息

组件目录中的睡眠定时器是使用带默认设置的睡眠定时器组件的原理图宏。睡眠定时器组件与一个中断组件（配置为默认设置）相连。

组件参数

将睡眠定时器原理图宏拖入设计中，双击睡眠定时器组件，打开 **Configure** 对话框。



睡眠定时器组件包含下列参数：

唤醒间隔

定义了睡眠定时器唤醒器件、生成中断（如果配置生成中断）的间隔，或定义实现该两个操作的间隔。接受的独立间隔有：2、4、8、16、32、64、128、256、512、1024、2048 以及 4096 ms。

在假设来自 ILO 的输入时钟为 1 kHz 的情况下，获取这些间隔值。实际上，根据 ILO 频率改变，睡眠定时器间隔将有所不同，如器件数据手册中所介绍的内容。

此参数定义初始配置。只有睡眠定时器停止时，软件才能重新配置该值。

使能睡眠定时器中断

该参数定义了经过选定间隔后，睡眠定时器组件是否生成中断。请注意，对于基于 ARM 的器件，需要通过中断唤醒 CPU。更多信息，请查阅 *《系统参考指南》* 中电源管理节中的内容。

此参数定义初始配置。软件可以重新配置该参数的设置。

时钟选择

睡眠定时器组件使用 CTW，并要求以 1 kHz 时钟频率进行操作。该时钟频率由内部低速振荡器（ILO）生成。ILO 1 kHz 时钟直接为 CTW 计数器提供脉冲。ILO 生成时钟时，没有使用外部组件，其功耗很低。

启动睡眠定时器的 API 函数自动启用 1 kHz 时钟。即使该组件停止后，API 函数仍保持该时钟的活动状态。第一个间隔的范围为 1 至（周期 + 1）毫秒。额外间隔发生于额定周期。

应用编程接口

通过应用编程接口（API）子程序，您可以使用软件对组件进行配置。下面的表格列出并说明了每个函数的接口。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“SleepTimer_1”分配给指定设计中组件的第一个实例。您可以将该实例重命名为符合标识符语法规则的任意唯一值。实例名称会成为每个全局函数名称、变量和符号常量的前缀。为了便于阅读，下表使用的实例名称为“SleepTimer”。

函数

函数	说明
SleepTimer_Start()	启动睡眠定时器操作。
SleepTimer_Stop()	停止睡眠定时器操作。
SleepTimer_EnableInt()	启用睡眠定时器组件，以在唤醒时生成中断。
SleepTimer_DisableInt()	禁用睡眠定时器组件，以在唤醒时生成中断。
SleepTimer_SetInterval()	设置睡眠定时器唤醒的间隔。
SleepTimer_GetStatus()	返回电源管理器中断状态寄存器的值，并清除该寄存器内的所有位。
SleepTimer_Init()	初始化和恢复随定制器提供的默认配置。
SleepTimer_Enable()	启用1 kHz ILO和CTW计数器。

全局变量

变量	说明
SleepTimer_initVar	指示睡眠定时器是否已初始化。变量将初始化为 0，并在第一次调用 SleepTimer_Start() 时设置为 1。这样，第一次调用 SleepTimer_Start()子程序后，组件不用重新初始化即可重启。 如果需要重新初始化此组件，则可在 SleepTimer_Start() 或 SleepTimer_Enable() 函数之前调用 SleepTimer_Init() 函数。

void SleepTimer_Start(void)

说明： 这是开始执行器件操作的首选方法。SleepTimer_Start()设置 initVar 变量，调用 SleepTimer_Init() 函数，然后调用 SleepTimer_Enable() 函数。启用 1 kHz ILO 时钟，并在睡眠定时器组件停止后，保持该时钟的活动状态。

参数： 无

返回值： 无

其他影响： 如果已设置 initVar 变量，则该函数仅调用 SleepTimer_Enable() 函数。

void SleepTimer_Stop(void)

说明： 停止睡眠定时器操作，禁用唤醒和中断。当 CTW 计数器达到终端计数时，器件将不唤醒，也不生成中断。

参数： 无

返回值： 无

其他影响： 停止睡眠定时器组件后，仍保持 1 kHz ILO 时钟的活动状态。

void SleepTimer_EnableInt(void)

说明： 启用 CTW 计数结束中断。

参数： 无

返回值： 无

其他影响： 无

void SleepTimer_DisableInt(void)

说明： 禁用 CTW 计数结束中断。

参数： 无

返回值： 无

其他影响： 无

void SleepTimer_SetInterval(uint8 interval)

说明： 设置 CTW 间隔周期。第一个间隔的范围为1至（周期 + 1）毫秒。额外间隔发生于额定周期。只有禁用 CTW 时，才能修改间隔值，亦即停止组件时，才能实现该操作。

参数： uint8 interval：CTW的间隔值

名称	值	额定周期
SleepTimer__CTW_2_MS	4'b0001	2 ms
SleepTimer__CTW_4_MS	4'b0010	4 ms
SleepTimer__CTW_8_MS	4'b0011	8 ms
SleepTimer__CTW_16_MS	4'b0100	16 ms
SleepTimer__CTW_32_MS	4'b0101	32 ms
SleepTimer__CTW_64_MS	4'b0110	64 ms
SleepTimer__CTW_128_MS	4'b0111	128 ms
SleepTimer__CTW_256_MS	4'b1000	256 ms
SleepTimer__CTW_512_MS	4'b1001	512 ms
SleepTimer__CTW_1024_MS	4'b1010	1024 ms
SleepTimer__CTW_2048_MS	4'b1011	2048 ms
SleepTimer__CTW_4096_MS	4'b1100	4096 ms

返回值： 无

其他影响： 无

uint8 SleepTimer_GetStatus(void)

说明： 返回睡眠定时器状态寄存器的状态，并清除正在挂起的状态位。唤醒后，应用程序代码必须调用该函数，以清除 ctw_int 状态位。不管已禁用还是已启用睡眠定时器的中断，用户代码必须调用该函数。

参数： 无

返回值： 如果发生相应的事件，将返回一个 8 位值（uint8），其中各位已被设置。下表中的常量表示的是该返回值可包含的两个事件的2位掩码。

常量	说明
SleepTimer_PM_INT_SR_ONEPPSP	发生了一个“one-pps”事件
SleepTimer_PM_INT_SR_CTW	发生了一个中央时轮（CTW）事件
SleepTimer_PM_INT_SR_FTW	发生了一个快速时轮（FTW）事件（请查阅器件数据手册，了解有关FTW事件的信息）。

其他影响： 如果没有给与SleepTimer相关的中断调用SleepTimer_GetStatus()函数，该中断将不被清除，并且当退出该中断时，又立即重新进入该中断。

睡眠定时器过期时，睡眠间隔将为0 ms，因为该中断一直被调用，直到通过 SleepTimer_GetStatus() 函数清除ctw_int标志为止。

如果在读取/清除寄存器时生成了一个中断，则该位保持不变（这将导致另一个中断）。

先报告，然后清除电源管理器中断状态寄存器中的所有中断状态位。某些位与该组件的操作无关。

唤醒后，（当禁用或启用睡眠定时器的中断时）应用程序代码必须调用该函数，以清除ctw_int状态位。发生CTW事件时，代码必须在1 ms（ILO 的一个时钟周期)内调用 SleepTimer_GetStatus()。

void SleepTimer_Init(void)

说明： 根据自定义程序“配置”对话框设置来初始化或恢复组件。无需调用SleepTimer_Init()，因为 SleepTimer_Start() API会调用此函数，这是开始组件操作的选用方法。设置CTW间隔周期，并启用或禁用CTW中断（根据定制器的设置情况）。

参数： 无

返回值： 无

其他影响： 无

void SleepTimer_Enable(void)

说明： 激活1 kHz ILO和 CTW并开始组件操作。无需调用SleepTimer_Enable()，因为SleepTimer_Start() API会调用此函数，这是开始组件操作的选用方法。

参数： 无

返回值： 无

其他影响： 无

MISRA 合规性

本节介绍了MISRA-C:2004合规性和本组件的偏差情况。定义了两种类型的偏差：

- 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

本节介绍了有关组件特定偏差的信息。*《系统参考指南》*的“MISRA 合规性”章节中介绍了项目偏差以及有关 MISRA 合规性验证环境的信息。

此睡眠定时器组件没有任何特定偏差。

固件源代码示例

PSoC Creator 在“Find Example Project”对话框中提供了包括原理图和代码示例的许多示例项目。要查看特定组件实例，请打开“Component Catalog”中的对话框或者原理图中的组件样例。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** 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”中的内容。

功能说明

睡眠定时器组件不影响器件进入低功耗模式。有关详细信息，请参见*《系统参考指南》*中“电源管理 API”一节介绍的内容。可以从 PSoC Creator 的帮助菜单中找到该指南。

睡眠定时器组件使用了一个中央时轮（CTW）。CTW 是一个以 1 kHz 频率自由运行的 13 位计数器，其时钟由 1 kHz ILO 提供。

请查阅器件数据手册，了解有关 CTW 和看门狗定时器（WDT）之间的关系的信息。

如前所述，可以将睡眠定时器的间隔分别配置为 2、4、8、16、32、64、128、256、512、1024、2048 以及 4096 ms。但是，请务必切记 Sleep Timer 的时钟源 ILO 的频率变化会影响 Sleep Timer 的间隔。此类变化详见器件数据手册。

为了正常操作睡眠定时器组件，每次唤醒器件以及每次生成睡眠定时器中断时，需要调用 SleepTimer_GetStatus()函数。

资源

睡眠定时器使用下面器件资源：

- 1 kHz ILO 时钟线
- CTW 计数器
- CTW 计数器的中断线

API 存储器大小

根据编译器、器件、所使用的 API 数量以及组件的配置情况不同，组件的存储器大小也不一样。下表提供了组件配置中所有 API 占用的存储器大小。

通过使用“释放”模式下的相应编译器，可以进行测量操作。在该模式下，存储器的大小得到优化。对于特定的设计，分析编译器生成映射文件后可以确定存储器的使用大小。

配置	PSoC 3 （Keil_PK51）		PSoC 5LP（GCC）	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
默认值	160	1	226	1

组件更改

本节列出了组件与先前版本相比的主要更改的内容。

版本	更改说明	更改原因/影响
3.20.b	Minor datahsheet edit.	
3.20.a	清除数据手册中有关PSoC 5的参考内容。	PSoC 5被替代为PSoC 5LP。
3.20	添加了MISRA合规性章节。	此组件没有任何特定偏差。
3.10	更改PSoC5LP的受支持的间隔组，以支持全范围间隔。	PSoC 5LP的电源管理使用模型不限制有效的睡眠时间。
	已添加了注意事项：对于基于ARM的器件，需要通过中断唤醒CPU。	
3.0	在PSoC 5中，受支持的间隔组时长被增大为4、8、16、32、64、28和256 ms。 更新了特性信息。添加了PSoC 5LP的存储器使用信息。	PSoC 5的电源管理使用模型可以部分释放有效的睡眠时间。更多有关信息，请参见*《系统参考指南》*中的内容。
	对数据手册进行了少量编辑。	提高可读性。
2.1	修正了SleepTimer_GetStatus()的执行：目前只有CTW中断状态被清除。	假设SleepTimer_GetStatus()函数是只清除CTW中断状态，因为组件使用了该定时器。清除其他中断状态是不准确的。 欲了解更多信息，请参考*《系统参考指南》*的“电源管理”一节中CyPmReadStatus()（该函数从SleepTimer_GetStatus()调用）的描述内容。
	在PSoC 5中，不会生成SleepTimer_SetInterval()不支持的间隔宏参数。	当生成不受支持的间隔宏时，可以消除混淆。
	如果选择了一个无效的间隔，会显示一个错误信息，并防止PSoC 5的器件关闭定制器。	不保存非准确数据。
	对定制器进行了少量的文本更改。	修复了几个拼写错误。
	已向组件调试器工具窗口添加了一些寄存器的描述内容。	增强了调试窗口支持。
2.0	对于PSoC 5，间隔限制为4、8或16 ms。	PSoC 5的电源管理使用模型会影响有效的睡眠时间。更多有关信息，请参考*《系统参考指南》*中的内容。
1.60.a	对数据手册进行了少量编辑和更新。	
1.60	纠正了Timewheel Configuration Register（时轮配置寄存器）2的覆盖问题。更新了源代码的批注。	消除寄存器的潜在覆盖问题，并提供了多个清除的批注。
	对数据手册进行了少量编辑和更新。	

版本	更改说明	更改原因/影响
1.50.a	在睡眠定时器组件的版本1.50中找到一个软件的缺陷。该缺陷可能覆盖共享的寄存器。在睡眠定时器组件的各个新版本中已经更新了该缺陷，因此不应在使用版本1.50。	
	向组件中添加了信息，以说明它与芯片修订版的兼容性。	如果组件在不兼容的芯片上使用，该工具将报告错误/警告。如果发生此情况，请更新到支持您的目标器件的修订版。
	对数据手册进行了少量编辑和更新。	
1.50	添加了Keil重入支持。	通过Keil编译器支持PSoC 3，以便能够从多个控制流调用函数。
	更改了API流：根据定制器的设置，SleepTimer_Start()将配置硬件。添加了SleepTimer_Init()函数。	所有组件应该有同样的执行流。为了能够更改组件的参数，需要调用SleepTimer_Stop()，并调用用于更改参数的各个函数，然后通过调用SleepTimer_Start()来重启组件。随后，为了恢复定制器的设置，组件被停止时需要将SleepTimer_initVar全局变量的值设置为 0 ，并重启组件。
	重新设计SleepTimer_Start（）函数，以便始终使能1 kHz的ILO时钟。以前该时钟已经被SleepTimer_Init()函数使能过一次。	停止组件操作和1 kHz的ILO时钟，解决潜在的问题，然后再次启动组件。
	添加了组件的XML说明。	这允许PSoC Creator提供一个机制来为该组件创建新的调试器工具窗口。
	优化了Microsoft Windows 7的自动滚动功能。	旨在避免出现不需要的滚动条。
1.10	移除了SleepTimer_Reset()函数，并添加了SleepTimer_GetStatus()函数。 在默认情况下，当某个设计中存在中断组件时，中断输出终端将连接至该中断组件。	进行多处更改，以解决功能并不全面的早期版本中存在的各种问题。

赛普拉斯半导体公司，2013-2016 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。