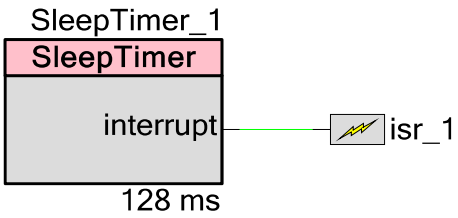
Sleep Timer
1.50

Features



- Wake up devices from low power modes: Alternate Active and Sleep
- Configurable option for issuing interrupt
- Generate periodic interrupts while device is in Active mode
- 12 discrete intervals: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096 ms

General Description

The Sleep Timer component is used to wake the device from Alternate Active and Sleep modes at a configurable interval. It can also be configured to issue an interrupt after wake up or while being in Active mode at a configurable interval.

When to use a Sleep Timer

The Sleep Timer component can be used to periodically wake up a device from Alternate Active and Sleep low power modes at a configurable interval, with or without issuing interrupts. It can also be used to generate periodic interrupts while the device is in Active mode – acting like a counter.

Periodic interrupts could also implemented by hardware counters. However, this would use hardware resources inefficiently and would require the device to remain in Active mode.

The Sleep Timer runs off of a single set of resources, so only one is available per design.

interrupt – Output

The Sleep Timer has one output connection – interrupt – and no input connections. The interrupt output contains the Central Time Wheel (CTW) interrupt source. The interrupt is issued when the CTW counter reaches the terminal count, specified in the component customizer or by API function.

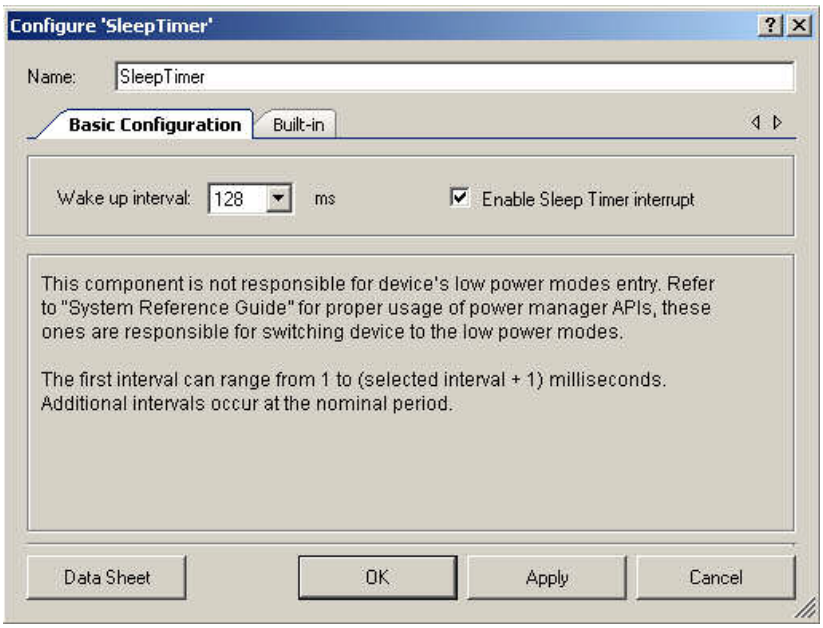
The output may be hidden on the symbol by de-selecting the **Enable Sleep Timer Interrupt** parameter.

Schematic Macro Information

The default Sleep Timer in the Component Catalog is a schematic macro using a Sleep Timer component with default settings. The Sleep Timer component is connected to an Interrupt component, which also is configured with default settings.

Parameters and Setup

Drag a Sleep Timer schematic macro onto your design and double-click it to open the Configure Sleep Timer dialog.



The Sleep Timer component contains the following parameters:

Wake up interval

Defines the interval at which the Sleep Timer will wake the device up and/or generate interrupts if it is configured to do so. Only discrete intervals are accepted: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096 ms.

This parameter defines an initial configuration. The software can reconfigure this value only when the Sleep Timer is stopped.

Enable Sleep Timer Interrupt

This parameter defines if the Sleep Timer component will issue an interrupt after the selected interval is reached. This parameter does not affect whether the component will wake up the device from the low power modes.

This parameter defines an initial configuration. The software can reconfigure this parameter's setting.

Clock Selection

The Sleep Timer component uses the CTW and requires a 1-kHz clock for its operation. This clock is produced by the internal low-speed oscillator (ILO).

The ILO 1-kHz clock feeds directly to the CTW counter. The ILO produces clocks with no external components, and with very low power consumption.

The API function that starts the Sleep Timer automatically enables the 1-kHz clock and leaves it enabled even after the component is stopped.

The first interval can range from 1 to (period + 1) milliseconds. Additional intervals occur at the nominal period.

Placement

There is no placement specific information.

Resources

The Sleep Timer uses the following device resources:

- 1-kHz ILO clock line
- CTW counter
- CTW counter’s interrupt line

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "SleepTimer_1" to the first instance of a component in a given design. You can rename the instance to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "SleepTimer".

Functions

Function	Description
void SleepTimer_Init(void)	Initializes/Restores default configuration provided with the customizer.
void SleepTimer_Enable(void)	Enables the 1 kHz ILO and the CTW counter.

Function	Description
void SleepTimer_Start(void)	Starts the Sleep Timer operation.
void SleepTimer_Stop(void)	Stops the Sleep Timer operation.
void SleepTimer_EnableInt(void)	Enables Sleep Timer component issuing interrupt on wake up.
void SleepTimer_DisableInt(void)	Disables Sleep Timer component issuing interrupt on wake up.
void SleepTimer_SetInterval(uint8)	Sets interval for Sleep Timer to wake up.
uint8 SleepTimer_GetStatus(void)	Returns value of Power Manager Interrupt Status Register and clears all bits in this register.

Global Variables

Variable	Description
SleepTimer_initVar	Indicates whether the Sleep Timer has been initialized. The variable is initialized to 0 and set to 1 the first time SleepTimer_Start() is called. This allows the component to restart without reinitialization after the first call to the SleepTimer_Start() routine. If reinitialization of the component is required, then the SleepTimer_Init() function can be called before the SleepTimer_Start() or SleepTimer_Enable() function.

void SleepTimer_Init (void)

Description: Initializes/Restores default configuration provided with the customizer. Sets CTW interval and enables or disables CTW interrupt (according to the customizer's settings).

Parameters: None.

Return Value: None.

Side Effects: None.

void SleepTimer_Enable(void)

Description: Enables the 1-kHz ILO and the CTW counter.

Parameters: None.

Return Value: None.

Side Effects: Enables the 1-kHz ILO clock and leaves it enabled after the Sleep Timer component is stopped.

void SleepTimer_Start(void)

Description: Starts the Sleep Timer operation. If this function is called, then the component will be initialized with values defined in the customizer. If parameters need to be changed, then stop component, change needed parameters by calling the desired API functions, and restart the component.

Parameters: None.

Return Value: None.

Side Effects: Enables the 1-kHz ILO clock and leaves it enabled after the Sleep Time component is stopped.

void SleepTimer_Stop(void)

Description: Stops the Sleep Timer operation: disables wake up and interrupt. The device will not wake up when the CTW counter reaches terminal count, nor will an interrupt be issued.

Parameters: None.

Return Value: None.

Side Effects: Leaves the 1-kHz ILO clock enabled after Sleep Time component is stopped.

void SleepTimer_EnableInt (void)

Description: Enables interrupt on the CTW terminal count reached.

Parameters: None.

Return Value: None.

Side Effects: None.

void SleepTimer_DisableInt (void)

Description: Disables interrupt on the CTW terminal count reached.

Parameters: None.

Return Value: None.

Side Effects: None.

void SleepTimer_SetInterval (uint8 interval)

Description: Sets CTW interval period. The first interval can range from 1 to (period + 1) milliseconds. Additional intervals occur at the nominal period.

Parameters: uint8 interval: interval's value for the CTW.

Name	Value	Period
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

Return Value: None.

Side Effects: Interval value can be only changed when the component is stopped (CTW is disabled).

uint8 SleepTimer_GetStatus (void)

Description: This function must always be called (when the Sleep Timer's interrupt is disabled or enabled) after wake up to clear the ctw_int status bit.

Parameters: None.

Return Value: Returns 8 bits value (uint8) with bits set if corresponding event has occurred.

Name	Description
SleepTimer_PM_INT_SR_ONEPPSP	A onepps event has occurred
SleepTimer_PM_INT_SR_CTW	A central time wheel event has occurred

Side Effects: If the SleepTimer_GetStatus() function is not called in an interrupt associated with the SleepTimer, the interrupt is not cleared and as soon as the interrupt is exited it will be re-entered.

Once the Sleep timer has expired, until the SleepTimer_GetStatus() function is called the sleep interval is functionally 0 ms, since SleepTimer_GetStatus clears the ctw_int bit.

If an interrupt gets generated at the same time as a register read/clear, the bit will remain set (which causes another interrupt). Reports and then clears all interrupt status bits in the Power Manager Interrupt Status Register. Some of the bits are not relevant to operation of this component.

This function must always be called (when the Sleep Timer's interrupt is disabled or enabled) after wakeup to clear the ctw_int status bit. SleepTimer_GetStatus() must be called within 1 ms (1 clock cycle of the ILO) after the CTW event occurred.

Sample Firmware Source Code

The following C language example demonstrates the basic functionality of the Sleep Timer component.

This example demonstrates device's wakeup from the Alternate Active Mode every 16 ms without issuing an interrupt. This example assumes the component has been placed in the schematic and renamed to "SleepTimer "

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>

void main()
{
    /* Enable all interrupts by the processor. */
    CYGlobalIntEnable;

    /* Configure and start SleepTimer */
    SleepTimer_Start();

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
```

```
        for(;;)
        {
            /* Place your code here to be executed in Active Mode.*/

            /* Switch to the Alternate Active Mode */
            CyWait();

           /* Code restarts here at wakeup, clear interrupt status bit */
            SleepTimer_GetStatus();
        }
    }
```

Functional Description

The Sleep Timer component is not responsible for the device's entry into low-power modes. Refer to the "Power Management APIs" section of the *System Reference Guide* for more information. The guide is available in the PSoC Creator's Help menu.

The Sleep Timer component uses Central Time Wheel (CTW). The CTW is a 1-kHz, free-running, 13-bit counter clocked by the 1-kHz ILO.

Refer also to the device data sheet to understand the relationship between the CTW and the Watch Dog Timer (WDT).

As described previously, the Sleep Timer can be configured to the following intervals: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, or 4096 ms. However, it is important to remember that it can have up to 20 percent deviation, because the CTW is source from the device's internal low speed oscillator.

For proper operation of the Sleep Timer component, you should call the SleepTimer_GetStatus function every time the device wakes up and every time the Sleep Timer interrupt is issued.

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data.

5.0-V/3.3-V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
Input					
Input Voltage Range	---		Vss to Vdd	V	
Input Capacitance	---		---	pF	
Input Impedance	---		---	Ω	
Maximum Clock Rate	---		67	MHz	

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.50.a	A firmware defect was found in version 1.50 of the SleepTimer component. This defect has the potential of overwriting shared registers. This defect has been fixed in later versions of the SleepTimer component so version 1.50 should not be used	
	Added information to the component that advertizes its compatibility with silicon revisions.	The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
	Minor datasheet edits and updates	
1.50	The Keil reentrancy support was added.	Support for PSoC 3 with the Keil compiler the capability for functions to be called from multiple flows of control.
	The API flow has been changed: `$SleepTimer`_Start() configures hardware according customizer's settings. The `$SleepTimer`_Init() function has been added.	All components should have the same execution flow. To change the component's parameters, the `$SleepTimer`_Stop() should be called, functions to change parameters should be called, and then component should be started again by calling `$SleepTimer`_Start(). To restore customizer's settings afterwards, the `$SleepTimer`_initVar global variable's value should be set to 0 (while component is stopped) and then started again.
	The `$SleepTimer`_Start() function was redesigned to always enable 1 kHz ILO clock. Previously, it was enabled once in the `$SleepTimer`_Init() function.	This fixes a potential issue when stopping component operation and the 1 kHz ILO, and then starting the component again.
	Added XML description of the component.	This allows for PSoC Creator to provide a mechanism for creating new debugger tool windows for this component.
	Optimized auto scroll for Microsoft Windows 7.	To avoid unneeded scroll bar appearing.
1.10	Removed SleepTimer_Reset() function and added SleepTimer_GetStatus() function. The interrupt output terminal is connected to an interrupt component by default when the component is placed in a design.	Various changes were made to fix issues with the previous version, which was not fully functional.

© Cypress Semiconductor Corporation, 2009-2010. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

PRELIMINARY