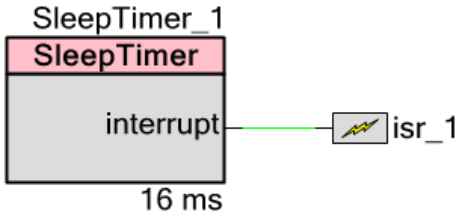
Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

Sleep Timer
3.20

Features



- Wakes up devices from low-power modes: Alternate Active and Sleep
- Contains configurable option for issuing interrupt
- Generates periodic interrupts while the device is in Active mode
- Supports twelve discrete intervals: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096 ms

General Description

The Sleep Timer component can be used to wake the device from Alternate Active and Sleep modes at a configurable interval. It can also be configured to issue an interrupt at a configurable interval.

When to Use a Sleep Timer

You can use the Sleep Timer component to periodically wake a device from Alternate Active and Sleep low-power modes at a configurable interval, with or without (PSoC 3 only) issuing interrupts. You can also use it to generate periodic interrupts while the device is in Active mode, like a counter.

Hardware counters can also implement periodic interrupts. However, this would use hardware resources inefficiently and would require the device to remain in Active mode.

The Sleep Timer uses a unique set of resources, so only one is available for each design.

interrupt – Output

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

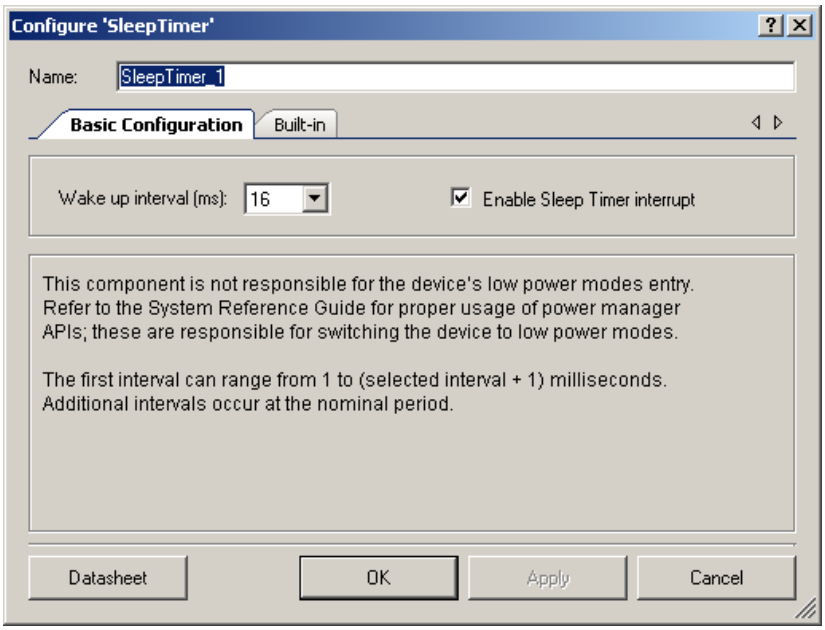
The output may be hidden on the symbol by deselecting the **Enable Sleep Timer Interrupt** parameter.

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.

Component Parameters

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



The Sleep Timer component contains the following parameters:

Wake up interval

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

These interval values assume a 1-kHz input clock from the ILO. In reality, the ILO's frequency, and thus the Sleep Timer interval, varies as described in the device datasheet.

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 whether the Sleep Timer component will issue an interrupt after the selected interval has elapsed. Note that an interrupt is required for the ARM-based devices for

CPU to wake up. Refer to the Power Management section in the *System Reference Guide* for details.

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.

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
SleepTimer_Start()	Starts Sleep Timer operation.
SleepTimer_Stop()	Stops Sleep Timer operation.
SleepTimer_EnableInt()	Enables the Sleep Timer component to issue an interrupt on wakeup.
SleepTimer_DisableInt()	Disables the Sleep Timer component to issue an interrupt on wakeup.
SleepTimer_SetInterval()	Sets the interval for the Sleep Timer to wake up.
SleepTimer_GetStatus()	Returns the value of the Power Manager Interrupt Status Register and clears all bits in this register.
SleepTimer_Init()	Initializes and restores the default configuration provided with the customizer.
SleepTimer_Enable()	Enables the 1-kHz ILO and the CTW counter.

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_Start(void)

Description: This is the preferred method to begin component operation. SleepTimer_Start() sets the initVar variable, calls the SleepTimer_Init() function, and then calls the SleepTimer_Enable() function. Enables the 1-kHz ILO clock and leaves it enabled after the Sleep Timer component is stopped.

Parameters: None

Return Value: None

Side Effects: If the initVar variable is already set, this function only calls the SleepTimer_Enable() function.

void SleepTimer_Stop(void)

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

Parameters: None

Return Value: None

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

void SleepTimer_EnableInt(void)

Description: Enables the CTW terminal count interrupt.

Parameters: None

Return Value: None

Side Effects: None

void SleepTimer_DisableInt(void)

Description: Disables the CTW terminal count interrupt.

Parameters: None

Return Value: None

Side Effects: None

void SleepTimer_SetInterval(uint8 interval)

Description: Sets the CTW interval period. The first interval can range from 1 to (period + 1) milliseconds. Additional intervals occur at the nominal period. You can only change the interval value when CTW is disabled, which you can do by stopping the component.

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

Name	Value	Nominal 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: None

uint8 SleepTimer_GetStatus(void)

Description: Returns the state of the Sleep Timer's status register, and clears the pending interrupt status bit. The application code must always call this function after wakeup to clear the ctw_int status bit. The code must call this function whether the Sleep Timer's interrupt is disabled or enabled.

Parameters: None

Return Value: Returns an 8-bit value (uint8) with bits set if a corresponding event has occurred. The constants shown in the following table describe the two-bit masks for the two events that this return value can contain.

Constant	Description
SleepTimer_PM_INT_SR_ONEPPSP	A one-pps event has occurred
SleepTimer_PM_INT_SR_CTW	A central time wheel event has occurred
SleepTimer_PM_INT_SR_FTW	A fast time wheel event has occurred (refer to the device datasheet for more information related to the FTW event).

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.

After the Sleep Timer has expired, the sleep interval is functionally 0 ms, because the interrupt will be called until the ctw_int flag is cleared by the SleepTimer_GetStatus() function.

If an interrupt is generated at the same time as a register read/clear, the bit remains 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 this component’s operation.

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

void SleepTimer_Init(void)

Description: Initializes or restores the component according to the customizer Configure dialog settings. It is not necessary to call SleepTimer_Init() because the SleepTimer_Start() API calls this function and is the preferred method to begin component operation. Sets CTW interval period 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:	Activates the 1-kHz ILO and the CTW and begins component operation. It is not necessary to call SleepTimer_Enable() because the SleepTimer_Start() API calls this function, which is the preferred method to begin component operation.
Parameters:	None
Return Value:	None
Side Effects:	None

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator components
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The SleepTimer component does not have any specific deviations.

Sample Firmware Source Code

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the “Find Example Project” topic in the PSoC Creator Help for more information.

Functional Description

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

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

See the device datasheet for information about the relationship between the CTW and the Watchdog 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 the Sleep Timer's clock source, the ILO, has frequency variation that will affect the Sleep Timer's interval. This variation is shown in the device datasheet.

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.

Resources

The Sleep Timer uses the following device resources:

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

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The following table provides the memory usage for all APIs available in the given component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Default	160	1	226	1

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
3.20.c	Minor datasheet edits.	
3.20.b	Minor datasheet edits.	
3.20.a	Edited datasheet to remove references to PSoC 5.	PSoC 5 has been replaced by the PSoC 5LP.
3.20	Added MISRA Compliance section.	The component does not have any specific deviations.
3.10	Changed the set of supported intervals for PSoC 5LP to support full range of intervals.	The Power Management usage model for PSoC 5LP has no limits for the available sleep times.
	Added note that interrupt is required for CPU to wake up in ARM-based devices.	
3.0	The set of supported intervals is enlarged to 4, 8, 16, 32, 64,128 and 256 ms selection for PSoC 5. Characterization information is updated. Added memory usage info for the PSoC 5LP.	The Power Management usage model for PSoC 5 partly releases the available sleep times. Refer to the *System Reference Guide* for more details.
	Minor datasheet edits.	Improve readability.
2.1	Fixed the implementation of SleepTimer_GetStatus(): only the CTW interrupt status is cleared now.	The SleepTimer_GetStatus() function is assumed to clear only CTW interrupt status as component uses this timer. Clearing other interrupt statuses is incorrect. Refer to the CyPmReadStatus() (this function is called from SleepTimer_GetStatus()) description in the Power Management section of the *System Reference Guide* for more details.
	The unsupported interval macros for the SleepTimer_SetInterval() parameter are not generated for PSoC 5.	Eliminate confusion when macros are generated for the intervals that are not supported.
	An error message is shown and customizer is prevented from closing for PSo C5 device if an invalid interval is selected.	Incorrect data must not be saved.
	Minor customizer text edits.	Fixed a few typos.
	Added a few more register descriptions to the component debugger tool window.	Enhanced debug window support.
2.0	Interval is restricted to 4, 8 or 16 ms selection for PSoC 5.	The Power Management usage model for PSoC 5 limits the available sleep times. Refer to the *System Reference Guide* for more details.
1.60.a	Minor datasheet edits and updates	

Version	Description of Changes	Reason for Changes / Impact
1.60	Fixed the Timewheel Configuration Register 2 clobbering issue. Updated the source code comments.	Eliminate potential register clobbering issues and provide more clear comments
	Minor datasheet edits and updates	
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.
	Changed the API flow: SleepTimer_Start() configures hardware according customizer's settings. Added the SleepTimer_Init() function.	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.
	Redesigned the SleepTimer_Start() function 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, 2012-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

