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.

PSoC 4 Low-Frequency Clock (cy_lfclk)

1.20

Features

- APIs to select Low-Frequency Clock (LFCLK) and Watch Crystal Oscillator (WCO) clock sources
- APIs to control Internal Low-Frequency Oscillator (ILO), WCO, Deep Sleep Timers, and Watchdog Timers (WDT)
- APIs to compensate and trim ILO frequency

General Description

The PSoC 4 Low-Frequency Clock (cy_lfclk) Component is a design-wide Component present in all PSoC 4 projects by default. It provides the application interface to configure various low-frequency clocks available in PSoC 4. These functions are not part of any Component libraries, but the functions may be used by them. The cy_lfclk Component also provides functions to configure WDTs and Deep Sleep Timers present in the device.

WDTs are available for PSoC 4100 BLE / PSoC 4200 BLE / PRoC BLE / PSoC 4200L / PSoC 4100M / PSoC 4200M / PSoC 4000S devices. Deep Sleep Timers are available for PSoC 4100S and PSoC Analog Coprocessor devices. The main functional difference between WDTs and Deep Sleep Timers is that Deep Sleep Timers cannot generate device reset but WDTs can.

The cy_lfclk is not visible in the Component Catalog, but the API library is available all the time.

When to Use cy_lfclk

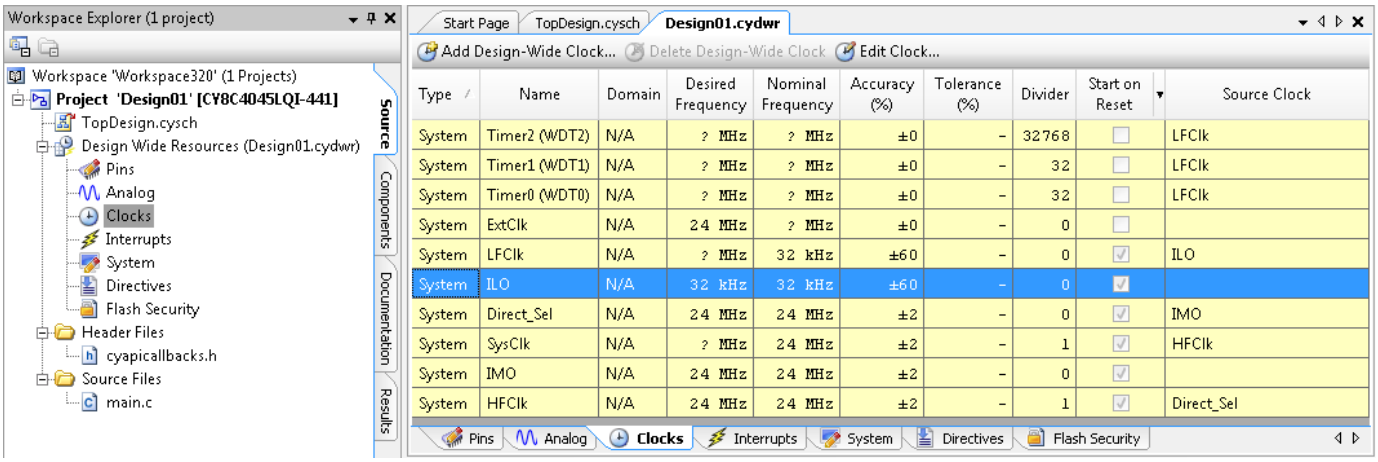
This Component provides an interface to configure low-frequency clocks and watchdog timers. Use this interface to configure these resources as needed. PSoC Creator uses the interface to initialize the resources as configured in the Design-Wide Resources (*<project>.cydwr*) file.

Input/Output Connections

The cy_lfclk Component does not have input or output connections.

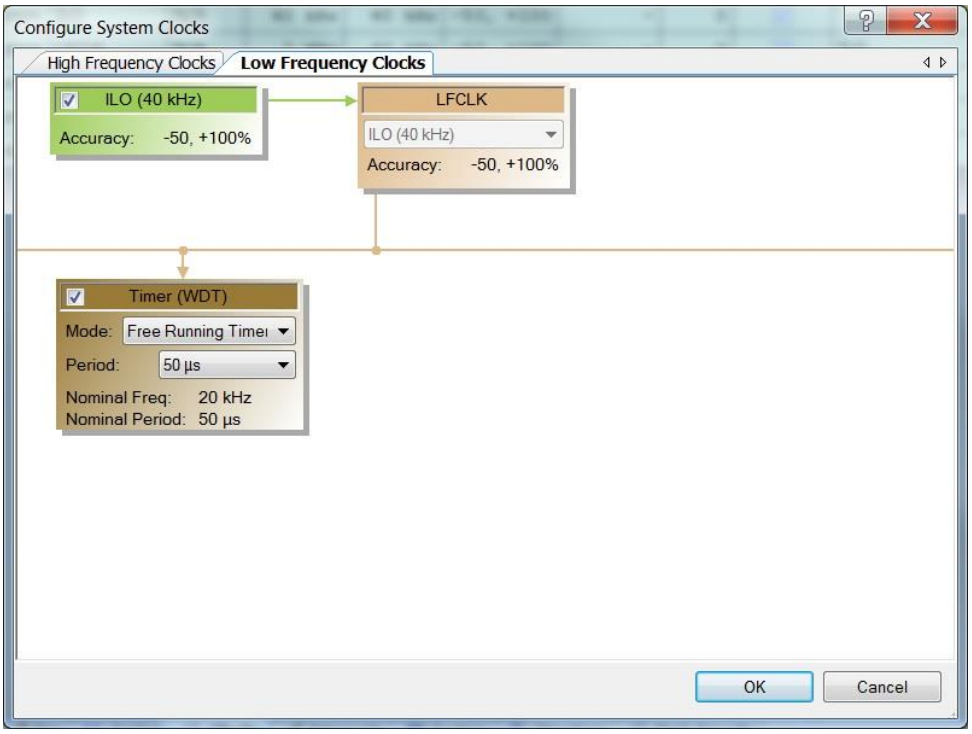
Component Parameters

In the PSoC Creator Workspace Explorer, double-click the **Clocks** node under the *<project>.cydwr* file to open the Clock Editor. Then, double-click any LFCLK clock source to open the Configure System Clocks dialog.

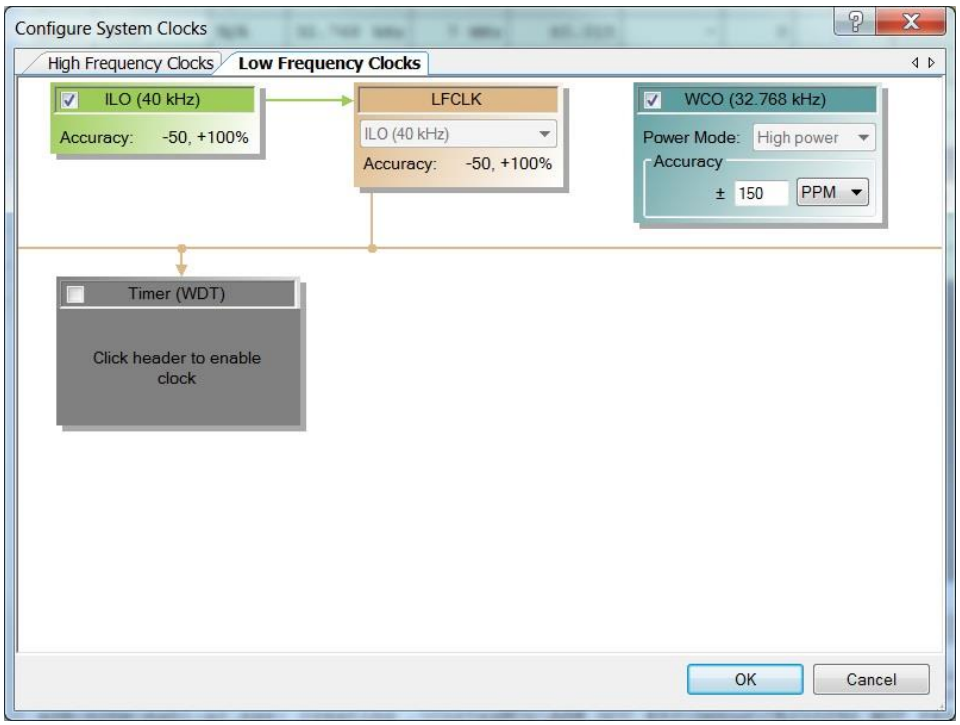


The Configure dialog has different options based on the device selected for the design.

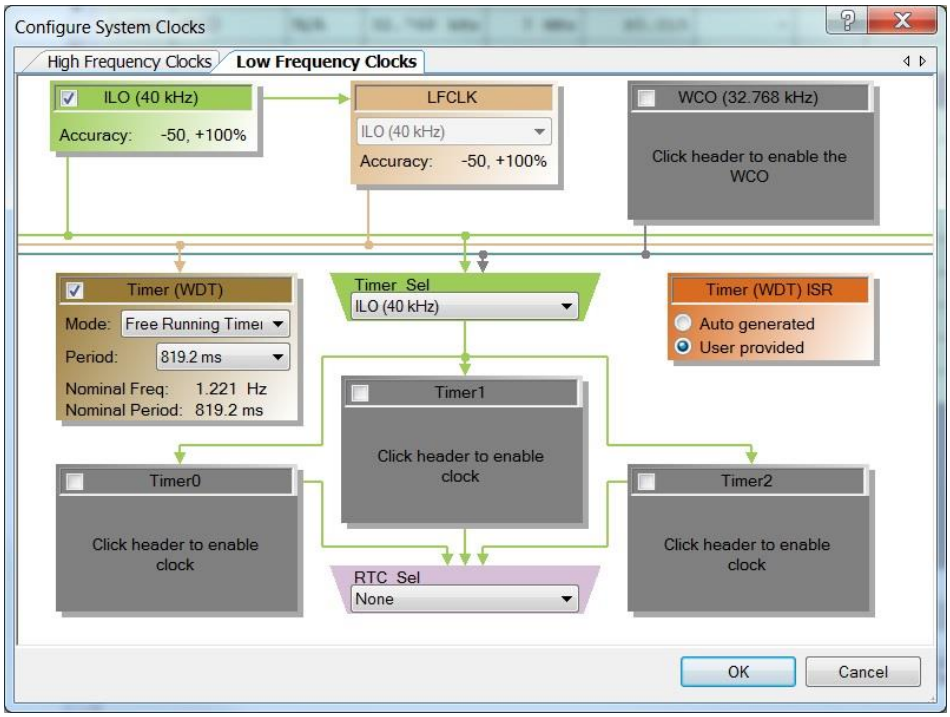
PSoC 4000 Configure Dialog



PSoC 4000S Configure Dialog

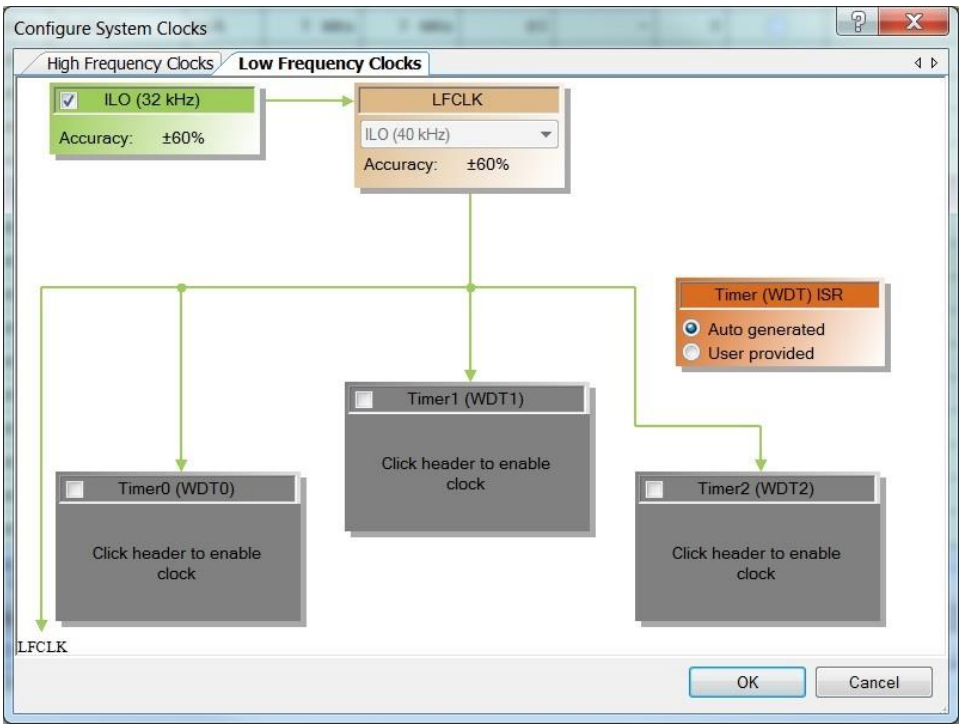


PSoC 4100S and PSoC Analog Coprocessor Configure Dialog

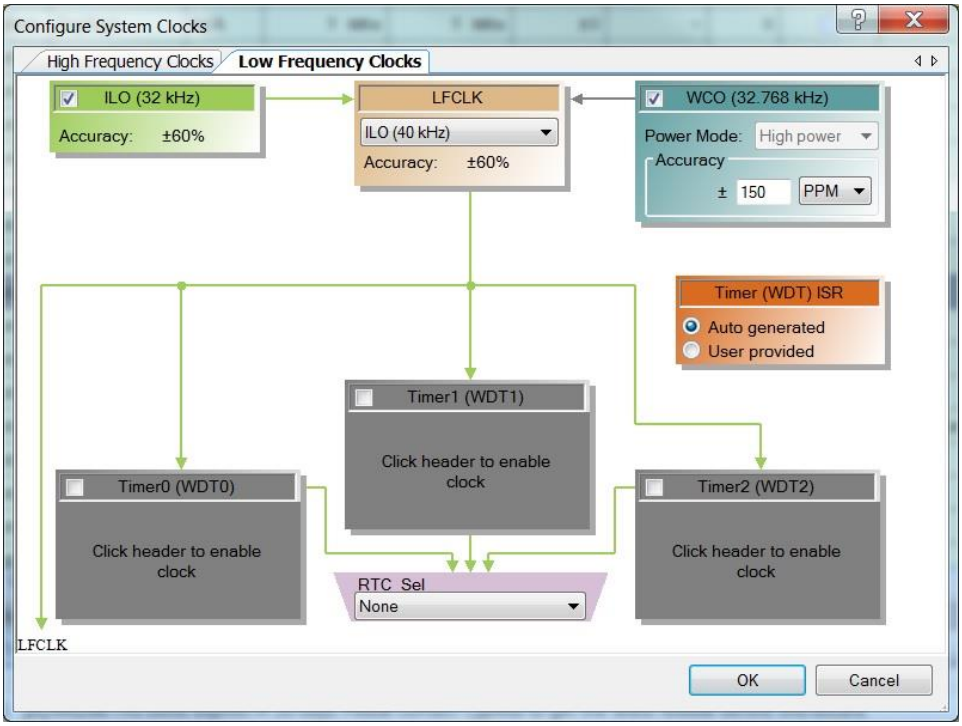


Note For the PSoC 4100S and PSoC Analog Coprocessor families, the Timer (WDT) ISR panel is used to configure the ISR for the Deep Sleep Timers.

PSoC 4100 / PSoC 4200 Configure Dialog



PSoC 4100 BLE / PSoC 4200 BLE / PRoC BLE /PSoC 4200L / PSoC 4100M / PSoC 4200M Configure Dialog



LFCLK Configuration Panels

The following table lists and describes the various panels that can be included in this dialog for various devices.

Panel	Description
ILO	This panel is used to configure the Internal Low-Frequency Oscillator. The ILO panel is available for all PSoC4 family devices.
LFCLK	This panel is used to select the LFCLK clock source. There are two possible clock sources: • ILO (32.000 kHz) • WCO (32.768 kHz) The option of selecting the LFCLK clock source WCO/ILO is applicable for PSoC 4100 BLE / PSoC 4200 BLE / PSoC 4100L / PSoC 4200L / PSoC 4100M / PSoC 4200M. For all other devices, this option is greyed out. **Warning** LFCLK clock source can be changed while program executing. It is important to switch off all peripherals that are driven by the LFCLK while clock source switching. At least the interrupts should be masked if it is impossible to switch off peripherals. If you do not switch off the peripherals driven by the LFCLK or do not mask the interrupts, it can prevent the program from halting. For PSoC 4000S / PSoC 4100S / PSoC Analog Coprocessor devices, LFCLK can only be sourced from ILO even though WCO is available. The LFCLK panel is available for all PSoC 4 family devices.
WCO	This panel is used to configure the Watch Crystal Oscillator. It provides the interface to the following settings: • WCO Power mode • WCO Accuracy The **Power mode** option is available only for PSoC 4 BLE devices. For all other PSoC 4 devices, this field is fixed to high power and is grayed out. The WCO panel is available for PSoC 4000S / PSoC 4100S / PSoC 4100 BLE / PSoC 4200 BLE / PRoC BLE / PSoC 4100L / PSoC 4200L / PSoC 4100M / PSoC 4200M / PSoC Analog Coprocessor family devices. This panel is not available for the PSoC 4000 / PSoC 4100 / PSoC 4200 devices.

Panel	Description
RTC_Sel	This panel is used to select which WDT or Deep Sleep Timers to use for RTC. It also provides a **None** option when not using WDT or Deep Sleep Timers to clock the RTC. There are four possible RTC clock sources: • None • Timer0 (WDT0) or Timer0 (Deep Sleep Timer0) • Timer1 (WDT1) or Timer1 (Deep Sleep Timer1) • Timer2 (WDT2) or Timer2 (Deep Sleep Timer2) The RTC_Sel mux is available for devices that have WDTs or Deep Sleep Timers. This panel is not available for the PSoC 4000 / PSoC 4100 / PSoC 4200 devices. WDTs are available for PSoC 4100 BLE / PSoC 4200 BLE / PRoC BLE / PSoC 4200L / PSoC 4100M / PSoC 4200M / PSoC 4000S devices. Deep Sleep Timers are available for PSoC 4100S and PSoC Analog Coprocessor devices.
Timer_sel	This mux selects the clock source for the Timers (Timer0/1/2) – ILO or WCO. For RTC support, it is recommended to enable WCO in the design and use that to source the Timers. **Warning** Timer_sel clock source can be changed while program executing. It is important to switch off all peripherals that are driven by Timer_sel while clock source switching. At least the interrupts should be masked if it is impossible to switch off the peripherals. If you do not switch off the peripherals driven by Timer_sel or do not mask the interrupts, it can prevent the program from halting. The Timer_sel mux is available only for PSoC 4100S and PSoC Analog Coprocessor devices.
WDT	This panel provides the option to enable and configure the Watchdog timers. The panel provides an interface to the following settings: Mode – WDT operation mode: • Free Running Timer – Does not generate an interrupt or reset. You can read the counter and set an interrupt in the firmware to generate occasional timing loops. • Periodic Timer – Generates an interrupt on a match event but no reset. The timer wraps at the set divider value. • Watchdog – Generates a reset on a match event (counter should be cleared before reaching a match event to prevent a reset). • Watchdog (w/interrupts) – Generates an interrupt on a match event and generates a reset on a 3rd unserviced interrupt. Period – This control provides the option to configure the WDT period. **Note** The WDT cascade options are not configurable using these panels but the APIs can be used to perform cascading of WDTs. WDT panel is available for all PSoC4 family devices.

Panel	Description
Timer0/Timer1/Timer2	This panel provides the option to enable and configure the Deep Sleep Timers. The panel provides an interface to the following settings: Mode – Timers operation mode: • Free Running Timer – Does not generate an interrupt or reset. You can read the counter and set an interrupt in the firmware to generate occasional timing loops. (Applicable only for Timer 0 and Timer 1) • Periodic Timer – Generates an interrupt on a match event but no reset. The timer wraps at the set divider value. The Deep Sleep Timer cascade options are not configurable using these panels but the APIs can be used to perform cascading of Timers. The Deep Sleep Timers panel is available only for PSoC 4100S and PSoC Analog Coprocessor devices.
Timer (WDT) ISR	This panel provides options for how the WDT or Deep Sleep Timer (depends on the device) interrupt handler should be implemented. For PSoC 4100 / PSoC 4200 / PSoC 4100 BLE / PSoC 4200 BLE / PRoC BLE / PSoC 4200L / PSoC 4100M / PSoC 4200M / PSoC 4000S family devices, this panel is used to configure the WDT interrupt. • If you select the auto-generated option, then the CySysWdtIsr() function is registered as the WDT interrupt handler. In this case, you can use the CySysWdtGetInterruptCallback()/ CySysWdtSetInterruptCallback() functions to Get/Set callbacks for each particular counter and use the CySysWdtEnableCounterIsr()/CySysWdtDisableCounterIsr()functions to Enable/Disable service of the registered callbacks for each particular counter. • If you select the User-provided option, then the CySysWdtIsr() function is not registered as the WDT interrupt handler. In this case, you can register either a custom handler or the CyWdtIsr() function by using the CyIntSetVector() API. For PSoC 4100S and PSoC Analog Coprocessor family devices, this panel is used to configure the Deep Sleep Timer interrupt. • If you select the auto-generated option, then the CySysTimerIsr() function is registered as the Deep Sleep Timer interrupt handler. In this case, you can use the CySysTimerGetInterruptCallback()/ CySysTimerSetInterruptCallback() functions to Get/Set callbacks for each particular counter and use the CySysTimerEnableIsr()/CySysTimertDisableIsr()functions to Enable/Disable service of the registered callbacks for each particular counter. • If you select the User-provided option, then the CySysTimerIsr() function is not registered as the Deep Sleep Timer interrupt handler. In this case, you can register either some custom handler or the CyTimerIsr() function by using the CyIntSetVector() API. For PSoC 4000 family devices, this panel is hidden by default. The WDT interrupt handler is implemented as user-provided.

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 The cy_lfclk Component is MISRA Compliant, except for the following specific deviations:

MISRA-C:2004 Rule	Rule Class (Required/ Advisory)	Rule Description	Description of Deviation(s)
12.4	R	Right hand operand of '&&' or '||' is an expression with possible side effects.	The reason of this violation that the one of operands is the value of register.
14.3	R	Before preprocessing, a null statement shall only occur on a line by itself; it may be followed by a comment provided that the first character following the null statement is a white-space character.	The issue is caused by the use of the CYASSERT macro, which is empty in RELEASE mode. There is no negative effect on this because the source is clear and readable.

API Memory Usage

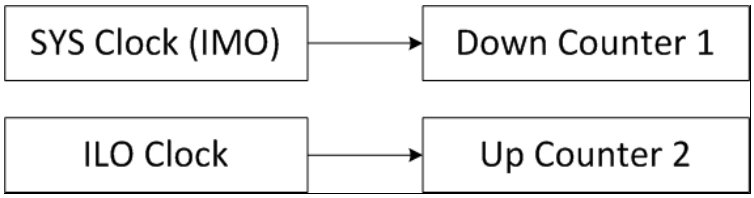
Refer to the PSoC 4 Family *System Reference Guide* (cy_boot).

ILO compensating / Trimming

Compensating / trimming processes use two-system counters controlled by two registers. Counter 1 is a down-counter clocked by the SysClk which is sourced by the accurate IMO. Counter 2 is an up-counter clocked by the ILO. The CySysClkIloStartMeasurement API configures these counters to be clocked respectively by the SysClk (Counter 1) and ILO (Counter 2).

Note SysClk should be sourced by IMO. If SysClk is sourced by another source (ECO or external source) the ILO compensating / trimming APIs can return unexpected results.

When a known 16-bit value (SysClk counts) is loaded into Counter 1, Counter 2 starts a clock from the ILO source. Counter 1 starts to count down on the SysClk, while Counter 2 starts to count up on the ILO source. Both counters stop when Counter 1 reaches 0, and the corresponding count value can be read out of Counter 2.



Using the ILO counts counted in Counter 2, the ILO current frequency can be calculated:

$$ILO_{CurrentFreq} = (ILO_Counts * SysClk_{Freq}) / SysClk_Counts$$

Basing on the obtained ILO frequency, the required number of the ILO cycles for a given delay can be calculated:

$$ILO_{accurate\ clocks} = (ILO_{CurrentFreq} * desiredDelay_clocks) / ILO_{NomFreq}$$

Also, the relative ILO accuracy can be calculated:

$$\Delta_{ILO\ accuracy} = (ILO_{CurrentFreq} - ILO_{NomFreq}) / ILO_{NomFreq}$$

Note The compensating API is applicable for all PSoC4 devices. The trimming API is applicable only for PSoC 4100 BLE / PSoC 4200 BLE / PSoC 4200L / PSoC 4100M / PSoC 4200M. For more details, see the API descriptions in the Application Programming Interface.

Warning The compensating/trimming functionality is based on the measured ILO frequency. The ILO frequency can be measured only in active-power mode. When switching the device into low-power modes (Sleep, DeepSleep), the ILO frequency might change and there is no possibility to measure the ILO frequency in low-power modes. Therefore, compensating/trimming functionality cannot guarantee +/-10% ILO accuracy in low-power modes.

Selecting WCO Output Source

For devices with WCO (PSoC 4100 BLE / PSoC 4200 BLE / PSoC 4200L / PSoC 4100M / PSoC 4200M / PSoC 4000S / PSoC 4100S / PSoC Analog Coprocessor) a possibility is created to externally drive the WCO. If you want to use an external pin to drive the WCO, the following procedure is required:

1. Disable the WCO.
2. Drive the wco_out pin to an external signal source.
3. Call CySysClkWcoClockOutSelect().

4. Enable the WCO and wait for 15 us before clocking the wco_out pin pad at a high potential [1].

Note Do not use the oscillator output clock prior to a 15-microsecond delay in your system.

If you want to use the WCO after using an external pin source:

1. Disable the WCO.
2. Drive the wco_out pin with external signal.
3. Call CySysClkWcoClockOutSelect().
4. Enable the WCO.

For stable WCO operation with an external clock source, the VDDA/VDDD external supply should be in the range from 1.65 V to 5 V. For stable WCO operation, the amplitude of the clock source driving the wco_out pin should be from a minimum of 1.0 V to a maximum of 1.6 V with respect to Vgnd because low-voltage devices depend on this external clock source. Therefore, the requirement is to provide an external clock source that toggles from 0 V to a minimum of 1.0 V or 0 V to the maximum of 1.6 V.

The specification for the WCO clock duty-cycle is 20% to 80%, so the external clock source should adhere to this same specification.

There are no limitations on the external clock frequency.

As mentioned in the procedure above, the wco_out pin should be used to drive an external clock source. Even if driving the wco_in pin would work, you get better performance using the wco_out pin because we can bypass the feedback resistor because this resistor causes an RC delay and causes duty-cycle variation.

In case when the WCO block is sourced by an external clock source it is possible to trim the IMO from the WCO only when the external clock source period is equal to the WCO period. Also, the external clock source accuracy should be higher or equal to the WCO accuracy.

See the CySysClkWcoClockOutSelect() explanation in the API section.

---

1 Assuming you are using the1.6 V clock amplitude, then the sequence would start at 1.6 V, then 0 V, then 1.6 V, etc. at a chosen frequency.

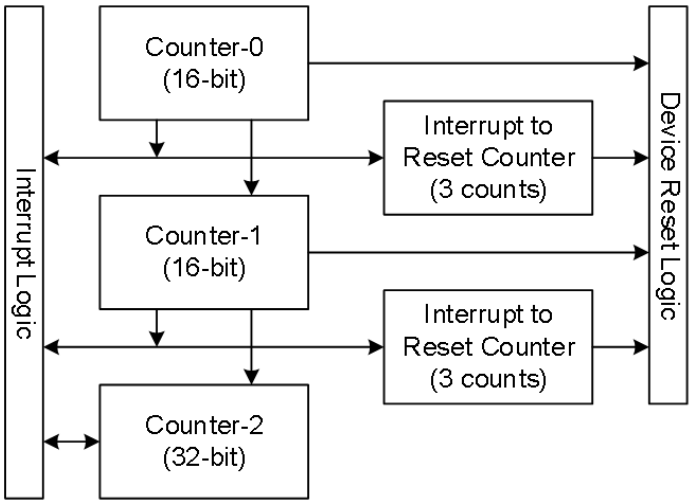
WDT Functional Description

PSoC 4100 / PSoC 4200 / PSoC 4100 BLE / PSoC 4200 BLE / PRoC BLE / PSoC 4200L / PSoC 4100M / PSoC 4200M

The WDT asserts an interrupt or a hardware reset to the device after a preprogrammed interval, unless it is periodically serviced in firmware. The WDT has two 16-bit counters (Counter-0 and Counter-1) and one 32-bit counter (Counter-2).

These counters can be configured to work independently or in cascade. The cascade configuration provides an option to increase the reset or interrupt interval.

If Counter-0 and Counter-1 are set in a cascade, it should be noted that Counter-0 is performing action on (match value + 1), but Counter-1 is performing action on (match value).



Counter-0 and Counter-1 generate an interrupt or a reset on reaching the specified terminal count for the first time. After three continuous unhandled interruptsCounter-0 and Counter-1 generate a reset, whereas Counter-2 only generates an interrupt. The interrupt must be cleared after entering the Interrupt Service Routine (ISR) in firmware by calling CySysWdtClearInterrupt() with the corresponding parameter.

Counter-0 and Counter-1 perform actions when the corresponding counter value equals the corresponding match value configured by calling CySysWdtWriteMode(). Counter-2 performs the action when the bit defined by calling CySysWdtWriteToggleBit() is toggled in Counter-2. For example, if the toggle bit is bit number 7 (configured by call to the CySysWdtWriteToggleBit(7) function), Counter-2 generates one interrupt per 2^7=128 WDT clocks.

Power Modes

In Active mode, an interrupt request from the WDT is sent to the CPU via IRQ 9. In Sleep or Deep Sleep power mode, the CPU subsystem is powered-down, so the interrupt request from the WDT is directly sent to the WakeUp Interrupt Controller (WIC), which will then wake up the

CPU. Then, the CPU acknowledges the interrupt request and executes the Interrupt Service Routine (ISR).

After waking from Deep Sleep, an internal timer value is set to zero until the ILO loads the register with the correct value. This led to an increase in Low-power mode current consumption. The work around is to wait for the first positive edge of the ILO clock before allowing the WDT_CTR_* registers to be read by CySysWdtReadCount() function.

Clock Source

The WDT is clocked by the LFCLK. The LFCLK can be sourced by a 32 kHz ILO or WCO. The WCO is available only for the PSoC 4100 BLE / PSoC4200 BLE, PSoC 4200L, and PSoC 4100M / PSoC 4200M devices. According to the device datasheet, the ILO accuracy is +/-60% over voltage and temperature. This means that the timeout period may vary by 60% from the configured value. Appropriate margins should be added while configuring WDT intervals to make sure that unwanted device resets do not occur on some devices.

According to the datasheet, the ILO accuracy can be obtained up to +/-10% by using a trimming API. Also, use a compensating API to obtain more accurate WDT functioning.

Refer to the device datasheet for more information on the oscillator accuracy.

Register Locking

Accidental corruption of the WDT configuration can be prevented by setting the bit-field WDT_LOCK of the CLK_SELECT register by calling the CySysWdtLock() function. When WDT is locked, any writing to the WDT_* and CLK_ILO* registers is ignored.

The CySysWdtUnlock() function should be called to allow WDT registers modification.

Clearing WDT

The LFCLK clock is asynchronous to the SYSCLK. So, generally, it takes 3 LFCLK cycles for the WDT register changes to come into effect. It is important to remember that a WDT should be cleared at least 4 cycles (3 + 1 for sure) before a timeout occurs, especially when small match values / low-toggle bit numbers are used.

The WDT counters should be cleared by calling the CySysWdtResetCounters() function with the parameter corresponding to the counters whose values are going to be cleared.

It is recommended to clear WDT counters from the portion of the code that is not directly associated with the WDT interrupt. It is possible that the main function of the firmware has crashed or is in an endless loop, but that the WDT interrupt vector is still intact and the WDT is getting serviced properly.

Reset Detection

The CySysGetResetReason() function can be used to detect if the watchdog has triggered a device reset.

Interrupt Configuration

The Global Signal Reference and Interrupt Components can be used for the ISR configuration. If the WDT is configured to generate an interrupt, the pending interrupts must be cleared within ISR (otherwise, the interrupt will be generated continuously):

A pending interrupt to the interrupt controller must be cleared by the call to the WDTISR_ClearPending() function, where WDTISR is the instance name of the interrupt Component.

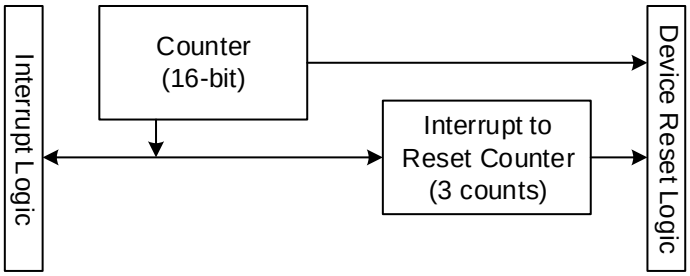
A pending interrupt to the WDT block must be cleared by the call to the CySysWdtClearInterrupt() function. The call to the function clears the unhandled WDT interrupt counter, if WDT is configured to be in "Generate interrupts and reset on 3rd unhandled interrupt" mode.

It is recommended to use the WDT ISR as a timer to trigger certain actions and to change the next WDT match value.

PSoC 4000 / PSoC 4000S / 4100S and PSoC Analog Coprocessor

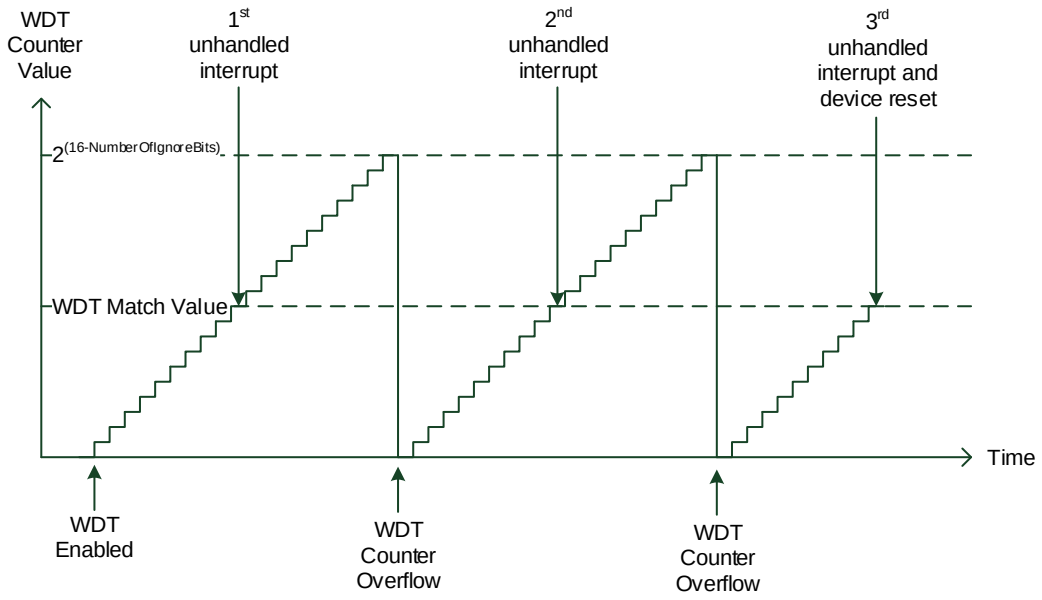
Note It is highly recommended to enable a WDT if the power supply can produce sudden brown-out events that may compromise the CPU functionality. This ensures that after a brown-out compromises the CPU functionality, the system always recovers.

The WDT asserts an interrupt or a hardware reset to the device after a preprogrammed interval, unless it is periodically serviced in firmware. The WDT is a 16- bit free-running up-counter.



The WDT generates an interrupt when the count value in the counter equals the configured match value.

It is important that the counter is not reset on a match. When the counter reaches a match value, it generates an interrupt and then keeps counting up till it overflows and rolls back to zero and reaches the match value again at which point another interrupt is generated.



To use a WDT for a periodic interrupt generation, the match value should be incremented in the ISR. As a result, the next WDT interrupt is generated when the counter reaches a new match value.

Also, some functionality is added to reduce the entire WDT counter period by specifying the number of most significant bits that are cut-off in the WDT counter. For example, if the CySysWdtWriteIgnoreBits() function is called with parameter 3, the WDT counter becomes a 13-bit free-running up-counter.

The WDT reset period can be calculated using the following equation:

$$WDT_{\mathrm{Re}\,setTime} = 2 * (\mathrm{LFCLK}_{Period} * (2^{(16 - NumberOfIgnoreBits)})) + (LFCLK_{Period} * WDT_{MatchValue})$$

Power Modes

In Active mode, the interrupt request from the WDT is sent to the CPU via IRQ 4. In the Sleep or Deep Sleep power mode, the CPU subsystem is powered down, so the interrupt request from the WDT is directly sent to the WakeUp Interrupt Controller (WIC). The WIC wakes up the CPU. Then, the CPU acknowledges the interrupt request and executes the Interrupt Service Routine (ISR).

Enabling or disabling a WDT requires three LFCLK cycles to come into effect. During that period the SYSCLK clock should be available. That means that the device should not be put into Deep Sleep mode during that period.

After waking from Deep Sleep, an internal timer value is set to zero until the ILO loads the register with the correct value. This leads to an increase in the Low-power mode current consumption. The workaround is to wait for the first positive edge of the ILO clock before allowing the WDT_CTR_* registers to be read by the CySysWdtReadCount() function.

Clock Source

The WDT is clocked by the LFCLK sourced by the 32 kHz ILO. The WDT reset must be disabled before disabling the ILO. Otherwise, any register write to disable the ILO is ignored. Enabling the WDT reset automatically enables the ILO.

According to the device datasheet, the ILO accuracy is +/-60% over voltage and temperature. This means that the timeout period may vary by 60% from the configured value. An appropriate margin should be added while configuring WDT intervals to make sure that unwanted device resets do not occur on some devices.

Use a compensating API to obtain more accurate WDT functioning. Refer to the device datasheet for more information on the oscillator accuracy.

Register Locking

This feature is not available for the device.

Clearing WDT

The LFCLK clock is asynchronous to the SYSCLK. So, generally, it takes three LFCLK cycles for the WDT registers changes to come into effect.

Note A WDT should be cleared at least for four cycles (3 LFCLK cycles + 1 to be sure) before a timeout occurs, especially when small match values / low toggle bit number are used.

It is recommended to clear WDT counters from the portion of the code that is not directly associated with a WDT interrupt. It is possible that the main function of the firmware has crashed or is in an endless loop, but that the WDT interrupt vector is still intact and the WDT is getting serviced properly.

Reset Detection

The CySysGetResetReason() function can be used to detect if the watchdog has triggered a device reset.

Interrupt Configuration

The Global Signal Reference and Interrupt Components can be used for the ISR configuration. If the WDT is configured to generate an interrupt, the pending interrupts must be cleared within ISR (otherwise, the interrupt will be generated continuously):

A pending interrupt to the interrupt controller must be cleared by the call to the WDTISR_ClearPending() function where WDTISR is the instance name of the interrupt Component.

A pending interrupt to the WDT block must be cleared by the call to the CySysWdtClearInterrupt() function. The call to the function will clear the unhandled WDT interrupt counter, if the WDT is configured to be in “Generate interrupts and reset on 3rd unhandled interrupt” mode.

It is recommended to use the WDT ISR as a timer to trigger certain actions and to change a next WDT match value.

Note If the Watchdog is configured as “Watchdog (w/ interrupts),” the interrupts from WDT are not passed to the CPU to avoid unregistered interrupts. On the third unhandled interrupt, a continuous device reset occurs. To avoid a continuous device reset, call the CySysWdtUnmaskInterrupt() API. After that, call the APIs with WDT interrupts handling/clearing.

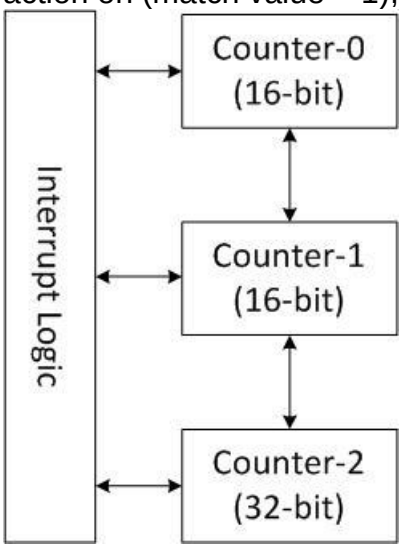
Deep Sleep Timers Functional Description

PSoC 4100S and PSoC Analog Coprocessor

The Deep Sleep Timer asserts an interrupt to the device after a preprogrammed interval, unless it is periodically serviced in firmware. The Timer has two 16-bit counters (Counter-0 and Counter-1) and one 32-bit counter (Counter-2).

These counters can be configured to work independently or in cascade. The cascade configuration provides an option to increase the reset or interrupt interval.

If Counter-0 and Counter-1 are set in a cascade, it should be noted that Counter-0 is performing action on (match value + 1), but Counter-1 is performing action on (match value).



Counter-0 and Counter-1 generate an interrupt on reaching the specified terminal count for the first time. All three counters can only generate an interrupt. The interrupt must be cleared after entering the Interrupt Service Routine (ISR) in firmware by calling CySysTimerClearInterrupt() with the corresponding parameter.

Counter-0 and Counter-1 perform actions when the corresponding counter value equals the corresponding match value configured by calling CySysTimerWriteMode(). Counter-2 performs the action when the bit defined by calling CySysTimerWriteToggleBit() is toggled in Counter-2. For example, if the toggle bit is bit number 7 (configured by call to the

CySysTimerWriteToggleBit(7) function), Counter-2 generates one interrupt per 2^7=128 Timer clocks.

Power Modes

In Active mode, an interrupt request from the Deep Sleep Timer is sent to the CPU. In Sleep or Deep Sleep power mode, the CPU subsystem is powered-down, so the interrupt request from the Deep Sleep Timer is directly sent to the WakeUp Interrupt Controller (WIC), which will then wake up the CPU. Then, the CPU acknowledges the interrupt request and executes the Interrupt Service Routine (ISR).

After waking from Deep Sleep, an internal timer value is set to zero until the ILO loads the register with the correct value. This led to increase in Low-power mode current consumption. The work around is to wait for the first positive edge of the ILO clock before allowing the WCO_WDT_CTR_* registers to be read by CySysTimerReadCount() function.

Clock Source

The Deep Sleep Timers are clocked by a 32 kHz ILO or WCO. According to the device datasheet, the ILO accuracy is +/-60% over voltage and temperature. This means that the timeout period may vary by 60% from the configured value if the Deep Sleep Timers are clocked by ILO. Appropriate margins should be added while configuring Timers intervals to make sure that unwanted interrupt was generated on some devices.

According to the datasheet, the ILO accuracy can be obtained up to +/-10% by using a trimming API. Also, use a compensating API to obtain more accurate Deep Sleep Timers functioning.

Refer to the device datasheet for more information on the oscillator accuracy.

Also to obtain accurate Deep Sleep Timers functioning use the WCO to drive Timers.

Clearing Deep Sleep Timers

The Deep Sleep Timer source (ILO or WCO) clock is asynchronous to the SYSCLK. So, generally, it takes 3 Timer source-cycles for the Deep Sleep Timers register changes to come into effect. It is important to remember that a Deep Sleep Timers should be cleared at least 4 cycles (3 + 1 for sure) before a timeout occurs, especially when small match values / low-toggle bit numbers are used.

The Deep Sleep Timer counters should be cleared by calling the CySysTimerResetCounters() function with the parameter corresponding to the counters whose values are going to be cleared.

It is recommended to clear Deep Sleep Timer counters from the portion of the code that is not directly associated with the Timer interrupt. It is possible that the main function of the firmware has crashed or is in an endless loop, but that the Timer interrupt vector is still intact and the Deep Sleep Timer is getting serviced properly.

Interrupt Configuration

The Global Signal Reference and Interrupt Components can be used for the ISR configuration. If the Deep Sleep Timer is configured to generate an interrupt, the pending interrupts must be cleared within ISR (otherwise, the interrupt will be generated continuously):

A pending interrupt to the interrupt controller must be cleared by the call to the TimerISR_ClearPending() function, where TimerISR is the instance name of the interrupt Component.

A pending interrupt to the Deep Sleep Timer block must be cleared by the call to the CySysTimerClearInterrupt() function.

It is recommended to use the Deep Sleep Timer ISR as a timer to trigger certain actions and to change the next Timer match value.

AC Electrical Characteristics

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted. Specifications are valid for 1.71 V to 5.5 V, except where noted.

Note Final characterization data for PSoC Analog Coprocessor devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.

Symbol	Description	Conditions	Min	Typ	Max	Units
$F_{ILOTRIM1}$	ILO frequency for PSoC 4100 / PSoC 4200 / PSoC 4100 BLE / PSoC 4200 BLE / PRoC BLE / PSoC 4200M / PSoC 4200L	Trimmed ILO frequency (+/ 60%)	15	32	50	kHz
	ILO frequency for PSoC 4000 / PSoC 4000S / 4100S / PSoC Analog Coprocessor		20	40	80	kHz
$F_{ILOTRIM2}$	ILO Frequency after trimming	ILO accuracy after successful completion of ILO trim	28.8	32	35.2	kHz
$T_{ILOTRIMDUR}$	Time taken to successfully complete ILO trim	IMO running at 24 MHz and using the APIs provided as part of cy_lfclk	0.5	-	30	ms
$T_{STARTILO}$	ILO Startup time		-	-	2	ms
F_{WCOTOL}	WCO Frequency tolerance	With 20 ppm crystal	-	50	250	ppm
C_{WCOL}	WCO Crystal load capacitance		6	-	12.5	pF
C_{WCO0}	WCO Crystal shunt capacitance		-	1.35	-	pF
R_{WCOESR}	WCO equivalent series resistance		-	50	-	kΩ

Symbol	Description	Conditions	Min	Typ	Max	Units
$T_{STARTWCO}$	WCO startup/settling time	WCO settling delay during System boot	-	-	500	ms
F_{WCO}	WCO frequency		-	32.768	-	kHz

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.20.b	Updated datasheet.	Added rule 14.3 to MISRA compliance table.
1.20.a	Minor datasheet edits.	
1.20	Updated the Component to use IP-block specific information instead of family devices for future device support. Edited datasheet.	 Added final characterization data for PSoC 4000S / PSoC 4100S devices
1.10.b	Edited datasheet.	Provided additional information about difference between WDTs and Deep Sleep Timers in General Description. Added information about compensating/trimming ILO frequency API.
1.10.a	Updated LFCLK Configuration Panels Added additional information about Deep Sleep Timers Changed term “WCO Timers” on “Deep Sleep Timers” instead	Provide more clear information
	Edited datasheet.	Final characterization data for PSoC 4000S, PSoC 4100S and PSoC Analog Coprocessor devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.
1.10	Added APIs to compensate and trim ILO frequency. Added API for externally driving the WCO. Added support for PSoC 4000S / PSoC 4100S / Analog Coprocessor Added WCO Timers APIs.	Increase ILO accuracy. To drive the WCO by an external source. Support new PSoC 4 device families. Support for WCO Timers.

Version	Description of Changes	Reason for Changes / Impact
1.0.b	Edited the datasheet.	Removed the Errata section. Fixed the PSoC 4000 Configure Dialog. The "Timer (WDT) ISR" panel and WDT interrupt generation is disabled when the Timer (WDT) panel is disabled.
		Added a note to Enable the WDT if the power supply might cause a brown-out event.
1.0.a	Added Component Errata section.	Document an issue and workaround with the PSOC 4000 WDT.
1.0	Initial Component version.	

© Cypress Semiconductor Corporation (and Infineon company), 2017-2021. 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.