PSoC 4 Low-Frequency Clock (cy_lfclk)

1.0

Features

- Included API to select the LFCLK clock source
- Included API to control the Internal Low Frequency Oscillator (ILO), Watch Crystal Oscillator (WCO), and Watchdog Timers
- Included GUI to control encountered features

General Description

The PSoC 4 Low-Frequency Clock (cy_lfclk) component provides the application interface to configure various low-frequency clocks available in PSoC 4. The functions are not part of any component libraries but may be used by them. The component also provides functions to configure watchdog timers present in the device. This a design-wide component, which is present by default in all PSoC 4 projects. This component is not visible in the Component Catalog, though the API library is available all the time.

When to Use a cy_lfclk

This component provides an interface to configure low-frequency clocks and watchdog timers. Use the interface to configure these resources as needed. PSoC Creator makes use of the interface to initialize the resources as configured in the Design-Wide Resources (.cydwr) file.

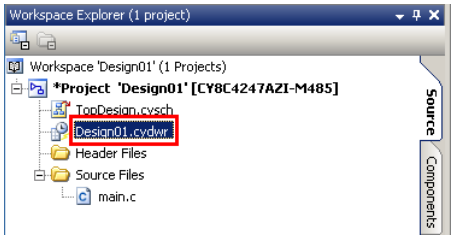
Input/Output Connections

The cy_lfclk component does not have input or output connections.

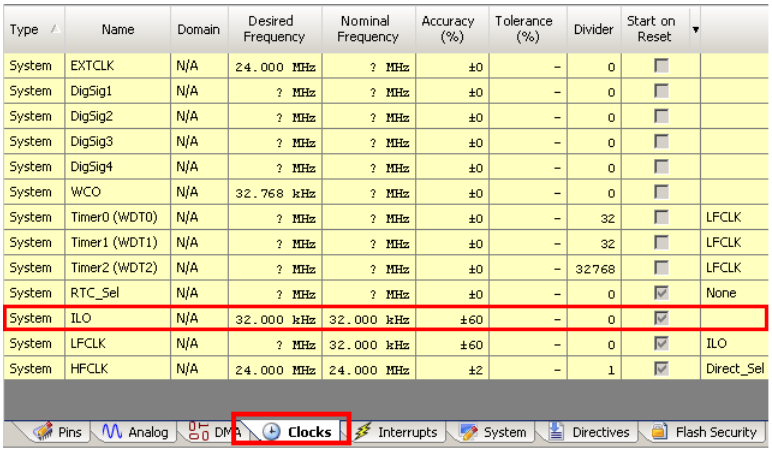
Component Parameters

In the PSoC Creator Workspace Explorer, double-click on the *<project>.cydwr* file to open the Design-Wide Resources (DWR) file. Then, click the **Clocks** tab to open the Clock Editor, and double click any LFCLK clock source to open the Configure System Clock dialog.

Workspace Explorer

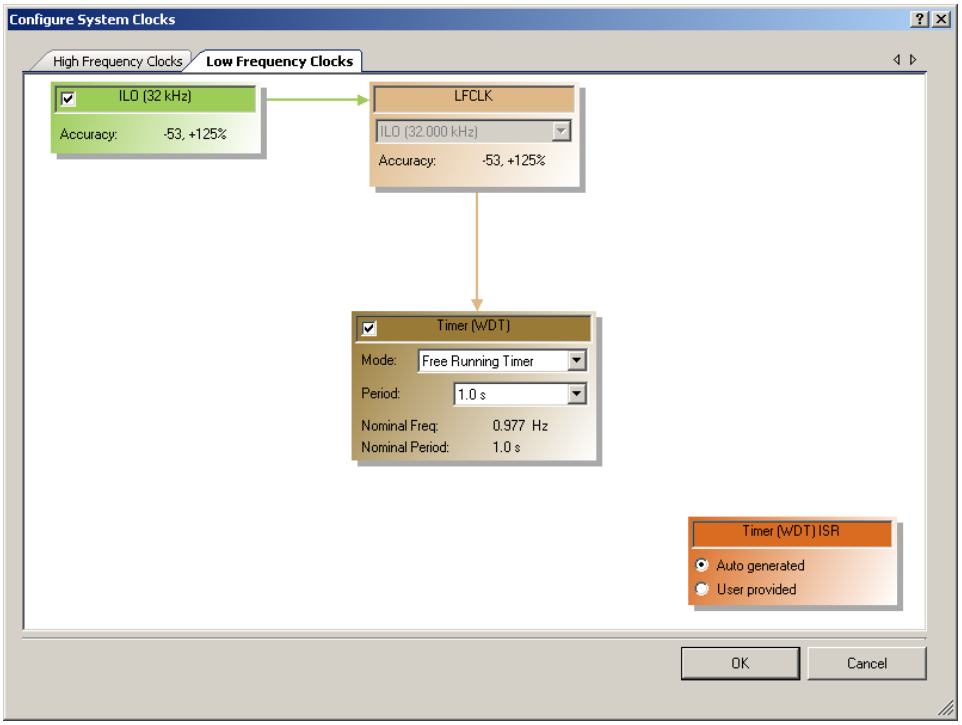


DWR Clock Editor

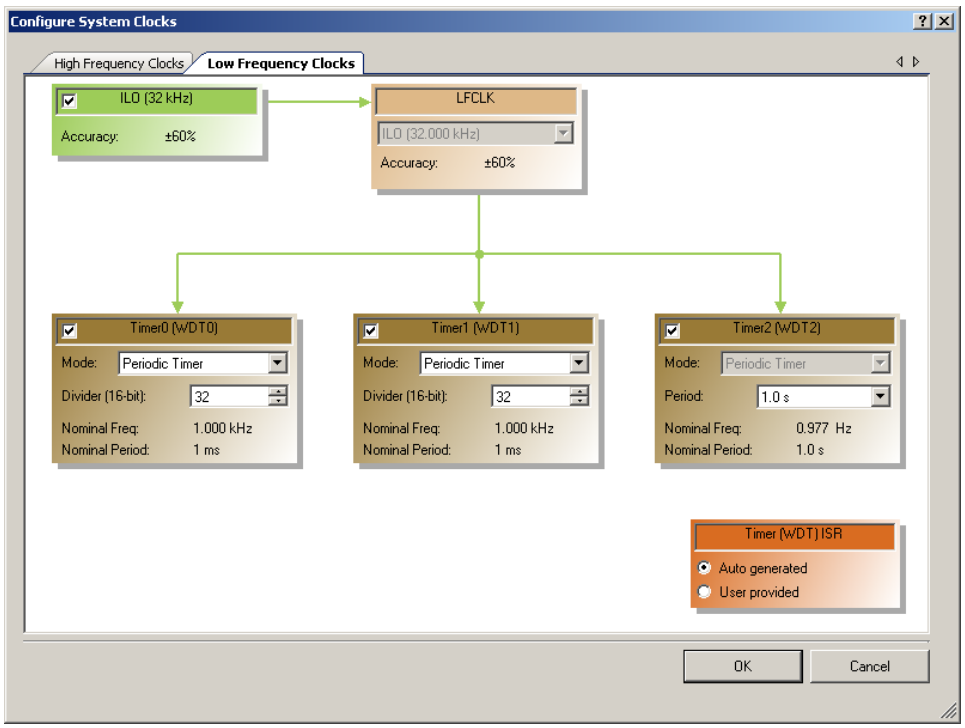


The Configure dialog has the different options based on the device selected for the design.

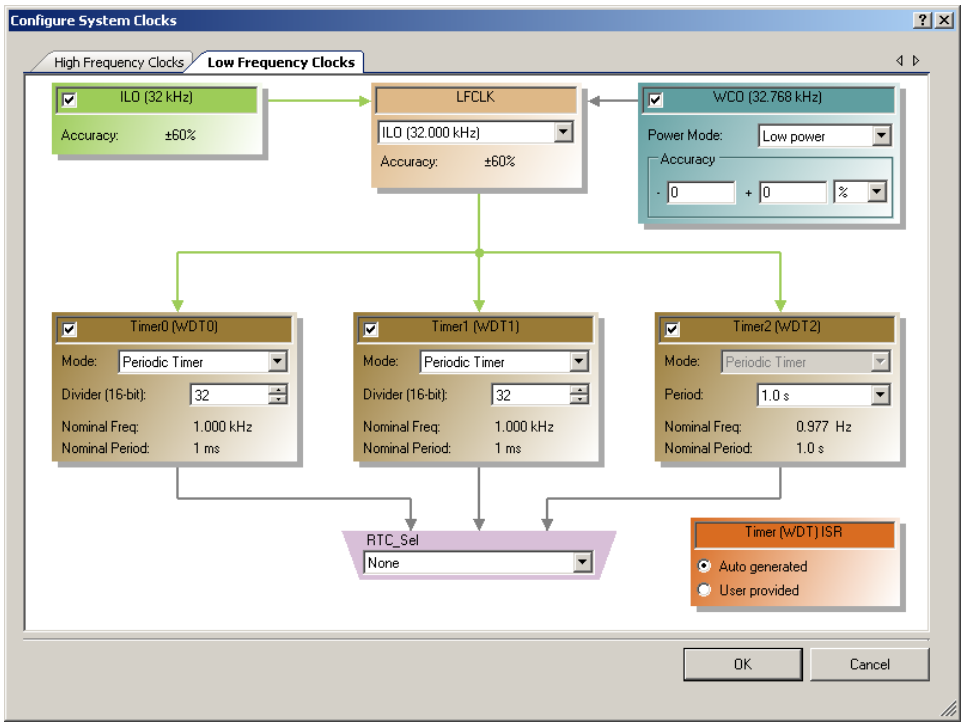
PSoC 4000 Configure Dialog



PSoC 4100/PSoC 4200 Configure Dialog



PSoC 4100 BLE/PSoC 4200 BLE/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.
LFCLK	This panel is used to select the LFCLK clock source. There are two possible clock sources: • ILO(32.000kHz) • WCO(32.768kHz)
WCO	This panel is used to configure the Watch Crystal Oscillator. Panel provides interface to the following settings: • WCO Power mode • WCO Accuracy **Note** The **WCO Power** Combo Box is not available for PSoC 4100 BLE and PSoC 4200 BLE devices.
RTC_Sel	This panel allows users to select which WDT timer to use for RTC. It also provides a **None** option when not using WDT to clock the RTC. There are four possible RTC clock sources: • None • Timer0(WDT0) • Timer1(WDT1) • Timer2(WDT2) **Note** This panel is not available for the PSoC 4000/PSoC 4100/PSoC 4200 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 – This does not generate an interrupt or reset; the user can read the counter and set an interrupt in the firmware togenerate occasional timing loops. • Periodic Timer – WDT1 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** WDT cascade options are not configurable using these panels but the APIs can be used to perform cascading of WDTs.

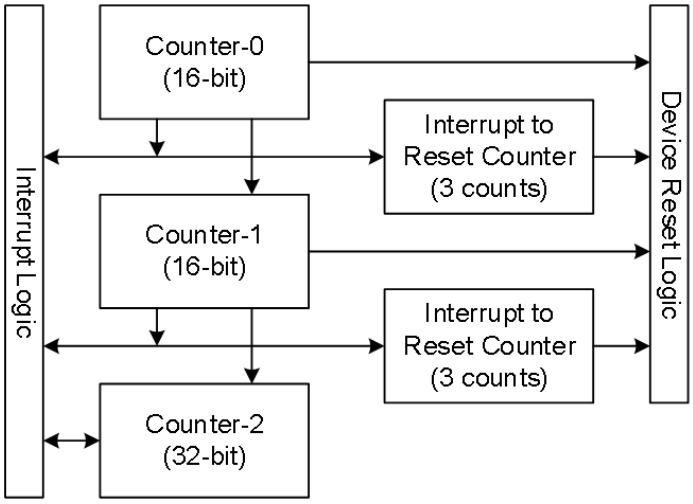
Panel	Description
Timer (WDT) ISR	This panel provides options for how the WDT interrupt handler should be implemented. If you select the auto-generated option, then the CySysWdtIsr() function is registered as the WDT interrupt handler. In this case, you can use the CySysWdtGetInterruptCallback()/ CySysWdtSetInterruptCallback() 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 some custom handler or the CyWdtIsr() function by using the CyIntSetVector() API.

Watchdog Timer Functional Description

PSoC 4100 / PSoC 4200 / PSoC 4100 BLE / PSoC 4200 BLE / PSoC4100M / PSoC4200M

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.



The Counter-0 and Counter-1 generate an interrupt or a reset on reaching the specified terminal count for the first time, or generate a reset after three continuous unhandled interrupt, 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 LFCLK. The LFCLK can be sourced by 32 kHz ILO or WCO. The WCO is available only for the PSoC 4100 BLE / PSoC4200 BLE 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. Appropriate margins should be added while configuring WDT intervals to make sure that unwanted device resets does not occur on some devices.

Refer to the device datasheet for more information on 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 registers changes to come into effect. This is important to remember that WDT should be cleared at least 4 cycles (3 + 1 for sure) before 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.

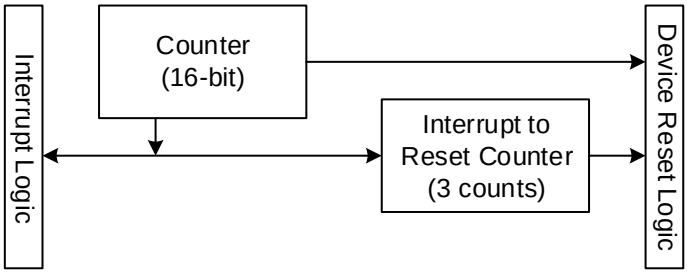
A pending interrupt to the WDT block must be cleared by the call to the CySysWdtClearInterrupt() function. The call to the function clears the unhandled WDT interrupt counter, if WDT is configured to be in "Generate interrupts and reset on 3rd unhandled interrupt" mode.

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

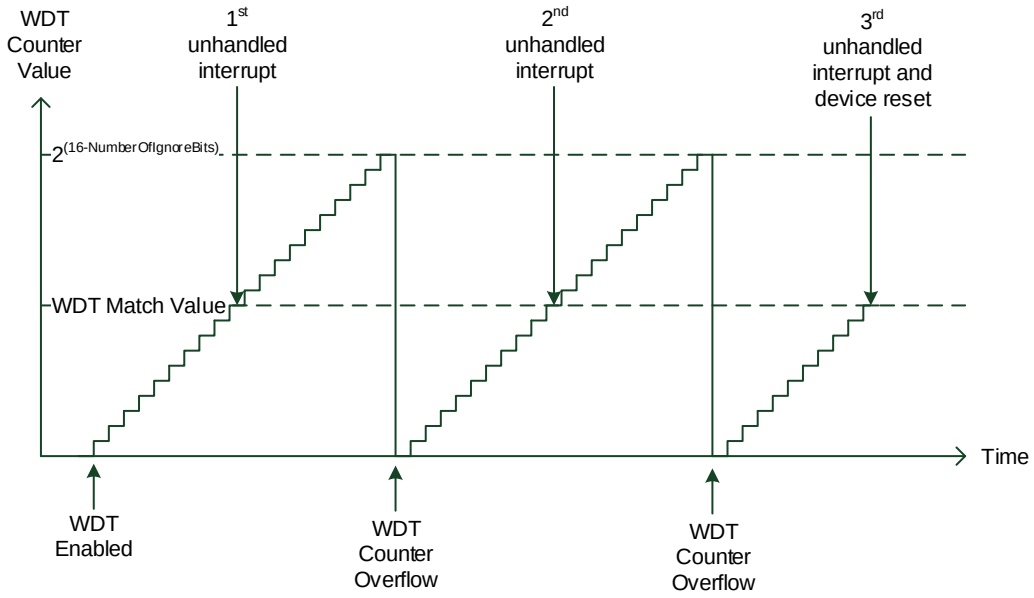
Note PSoC Creator automatically disables the WDT on startup. It is highly recommended to enable the WDT if the power supply might produce sudden brown-out events, which may compromise the CPU functionality. Enabling the WDT ensures that the system always recovers after a brown-out compromises the CPU functionality. To enable the WDR, either set it via the **Clocks** tab in the *<project>.cydwr* file or place a Global Signal Reference component in the design.

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.



If you want 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 the new match value.

Also some functionality is added to reduce 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 called with parameter 3, the WDT counter becomes a 13-bit free-running up-counter.

The WDT reset period can be calculated using 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), which then 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 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 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 will automatically enable 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. An appropriate margin should be added while configuring WDT intervals to make sure that unwanted device resets do not occur on some devices.

Refer to the device datasheet for more information on 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 WDT is configured to be in “Generate interrupts and reset on 3rd unhandled interrupt mode.

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

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using the 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 "cy_lfclk" to the first instance of a component in a given design. You can rename it 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 "cy_lfclk".

Functions

ILO ConfigurationFunction	Description
CySysClkIloStart()	Enables the ILO.
CySysClkIloStop()	Disables the ILO.

void CySysClkIloStart(void)

Description: Starts the ILO. Refer to the device datasheet for the ILO startup time.

Parameters: None.

Return Value: None.

Side Effects and Restrictions: None.

void CySysClkIloStop(void)

Description: Disables the ILO.

Parameters: None.

Return Value: None.

Side Effects and Restrictions: This function will have no effect if the WDT is locked (CySysWdtLock() is called). Call CySysWdtUnlock() to unlock the WDT and be able to stop ILO.

PSoC 4100 / PSoC 4200: The ILO is required for the WDT operation.

PSoC 4100 BLE / PSoC 4200 BLE / PSoC 4100M / PSoC4200M: Stopping the ILO affects the peripheral clocked by the LFCLK, if the LFCLK is configured to be sourced by the ILO.

LFCLK Source Configuration (PSoC 4100 BLE/PSoC 4200 BLE/PSoC 4200M)

void CySysClkSetLfclkSource(uint32 source)

Description: Sets the clock source for the LFCLK clock.

The switch between LFCLK sources must be done between positive edges of the LFCLK, because the glitch risk is around LFCLK positive edge. To ensure that the switch can be done safely, the WDT counter value is read until it changes.

That means that the positive edge just finished and the switch is performed. The enabled WDT counter is used for that purpose. If no counters are enabled, the counter 0 is enabled. And after LFCLK source is switched, the counter 0 configuration is restored back.

The function is applicable only for the devices with the WCO support.

Parameters: source: One of the available LFCLK sources.

Define	Description
CY_SYS_CLK_LFCLK_SRC_ILO	Internal Low Frequency (32 kHz) Oscillator (ILO)
CY_SYS_CLK_LFCLK_SRC_WCO	Low Frequency Watch Crystal Oscillator (WCO)

Return Value: None.

Side Effects and Restrictions: This function will have no effect if the WDT is locked (CySysWdtLock() is called). Call CySysWdtUnlock() to unlock the WDT.

The current source and the new source must both be running and stable before calling this function.

Changing the LFCLK clock source may change the LFCLK clock frequency and affect the functionality that uses this clock. For example, watchdog timer (WDT) is clocked by LFCLK.

The LFCLK must be sourced by the WCO for the ECO startup, if Deep Sleep mode operation is required for the BLE component. If no Deep Sleep mode operation is required for the BLE component, the LFCLK can be sourced by either ILO or WCO.

WCO configuration (PSoC 4100 BLE/PSoC 4200 BLE/PSoC 4100M/PSoC 4200M)

Function	Description
CySysClkWcoStart()	Enables 32KHz Crystal Oscillator.
CySysClkWcoStop()	Disables 32KHz Crystal Oscillator.
CySysClkWcoSetPowerMode()	Sets Power mode for the 32 KHz WCO.
CySysClkWcoClockOutSelect()	Selects WCO block output.

void CySysClkWcoStart(void)

Description: Enables the Watch Crystal Oscillator (WCO). The WCO is used as a source for LFCLK. Similar to ILO, WCO is also available in all modes except Hibernate and Stop modes.

The WCO is always enabled in High Power Mode (HPM). Refer to the device datasheet for the WCO startup time. Once WCO becomes stable it can be switched to Low Power Mode (LPM). Note that oscillator can be unstable during a switch and hence its output should not be used at the moment.

The CySysClkWcoSetPowerMode() function configures WCO power mode.

Parameters: None.

Return Value: None.

void CySysClkWcoStop(void)

Description: Disables the 32KHz Crystal Oscillator.

Parameters: None.

Return Value: None.

uint32 CySysClkWcoSetPowerMode(uint32 mode)

Description: Sets Power mode for the 32 KHz WCO.

Parameters: uint32 mode:

Define	Description
CY_SYS_CLK_WCO_HPM	High power mode
CY_SYS_CLK_WCO_LPM	Low power mode

Return Value: Previous Power mode. The same as parameters.

void CySysClkWcoClockOutSelect(uint32 clockSel)

Description: Selects the WCO block output. In addition to generating 32.768 kHz clock from external crystals, WCO block can output external clock input on wco_in pin to LFCLK selection. The API lets you select between the sources – External crystal or external pin.

Parameters: clockSel:

Define	Description
CY_SYS_CLK_WCO_HPM	High power mode
CY_SYS_CLK_WCO_LPM	Low power mode

WDT configuration (PSoC 4100/PSoC 4200/PSoC 4100 BLE/PSoC 4200 BLE/ PSoC 4100M/PSoC 4200M)

Function	Description
CySysWdtEnable	Enables the specified WDT counters. All the counters specified in the mask are enabled.
CySysWdtDisable	Disables the specified WDT counters. All the counters specified in the mask are disabled. The function waits for the changes to come into effect.
CySysWdtGetEnabledStatus	Reads the enabled status of one of the three WDT counters.
CySysWdtSetMode	Writes the mode of one of the three WDT counters.
CySysWdtGetMode	Reads the mode of one of the three WDT counters.
CySysWdtSetClearOnMatch	Configures the WDT counter Clear On Match setting. If configured to Clear On Match , the counter counts from 0 to MatchValue giving it a period of (MatchValue + 1).
CySysWdtGetClearOnMatch	Reads the Clear On Match setting for the specified counter.
CySysWdtSetCascade	Writes the two WDT cascade values based on the combination of mask values specified.
CySysWdtGetCascade	Reads the two WDT cascade values returning a mask of the bits set.
CySysWdtSetMatch	Configures the WDT counter match comparison value.
CySysWdtGetMatch	Reads the WDT counter match comparison value.
CySysWdtSetToggleBit	Configures which bit in WDT counter 2 to monitor for a toggle. When that bit toggles, an interrupt is generated if the mode for counter 2 has enabled interrupts.
CySysWdtGetToggleBit	Reads which bit in WDT counter 2 is monitored for a toggle.
CySysWdtLock	Locks out configuration changes to the Watchdog timer registers and ILO configuration register.
CySysWdtUnlock	Unlocks the Watchdog Timer configuration register.
CySysWatchdogFeed	
CySysWdtGetCount	Reads the current WDT counter value.
CySysWdtResetCounters	Resets all the WDT counters set in the mask.
CySysWdtGetInterruptSource	Reads a mask containing all the WDT counters interrupts that are currently set by the hardware, if a corresponding mode is selected.
CySysWdtClearInterrupt	Clears all the WDT counter interrupts set in the mask. Calling this function also prevents from Reset when the counter mode is set to generate 3 interrupts and then the device resets. All the WDT interrupts are to be cleared by the firmware, otherwise interrupts are generated continuously.
CySysWdtSetInterruptCallback	Sets the ISR callback function for the particular WDT counter. These functions are called on the WDT interrupt.

Function	Description
CySysWdtGetInterruptCallback	Gets the ISR callback function for the particular WDT counter.
CySysWdtEnableCounterIsr	Enables the ISR callback servicing for the particular WDT counter.
CySysWdtDisableCounterIsr	Disables the ISR callback servicing for the particular WDT counter.
CySysWdtIsr	This is the handler of the WDT interrupt in CPU NVIC. The handler checks which WDT is triggered in the interrupt and calls the respective callback functions configured by the user by using CySysWdtSetIsrCallback() API. The order of the callback execution is incremental. Callback-0 is run as the first one and callback-2 is called as the last one.
CySysTimerDelay	These functions implement the delay specified in the LFCLK clock ticks. The specified WDT counter should be configured as described below and started.
CySysTimerDelayUntilMatch	

void CySysWdtEnable(uint32 counterMask)

Description: Enables the specified WDT counters. All the counters specified in the mask are enabled.

Parameters: counterMask: Mask of all counters to enable:

Define	Counter
CY_SYS_WDT_COUNTER0_MASK	0
CY_SYS_WDT_COUNTER1_MASK	1
CY_SYS_WDT_COUNTER2_MASK	2

Return Value: None.

Side Effects and Restrictions: Enabling or disabling a WDT requires 3 LFCLK cycles to come into effect. Therefore, the WDT enable state must not be changed more than once in that period.

After WDT is enabled, it is illegal to write WDT configuration (WDT_CONFIG) and control (WDT_CONTROL) registers. That means that all WDT functions that contain ‘write’ in the name (with the exception of CySysWdtWriteMatch() function) are illegal to call once WDT enabled.

- **PSoC 4100 / PSoC 4200:** This function enables the ILO.
- **PSoC 4100 BLE / PSoC 4200 BLE / PSoC 4100M / PSoC4200M:** The LFLCK should be configured before calling this function. The desired source should be enabled and configured to be the source for the LFCLK.

void CySysWdtDisable(uint32 counterMask)

Description: Disables the specified WDT counters. All counters specified in the mask are disabled. The function waits for the changes to come into effect.

Parameters: counterMask: Mask of all counters to disable:

Define	Counter
CY_SYS_WDT_COUNTER0_MASK	0
CY_SYS_WDT_COUNTER1_MASK	1
CY_SYS_WDT_COUNTER2_MASK	2

Return Value: None.

uint32 CySysWdtGetEnabledStatus (uint32 counterNum)

Description: Reads the enabled status of one of the three WDT counters.

Parameters: counterNum: The valid range [0-2]. Number of the WDT counter.

Return Value: Status of the WDT counter:

0 - counter is disabled, 1 - counter is enabled

Side Effects and Restrictions: This function returns the actual WDT counter status from the status register. It may take up to 3 LFCLK cycles since the WDT counter was enabled for the WDT status register to contain actual data.

void CySysWdtSetMode(uint32 counterNum, uint32 mode)

Description: Writes the mode of one of the three WDT counters. If WDT mode is not configured, WDT timers are in a free running mode.

Parameters: counterNum: The valid range [0-2]. Number of the WDT counter.

mode: Mode of operation for the counter:

Define	Mode
CY_SYS_WDT_MODE_NONE	Free running
CY_SYS_WDT_MODE_INT	Interrupt generated on a match for Counter 0 and 1, and on a bit toggle for Counter 2
CY_SYS_WDT_MODE_RESET	Reset on a match (valid for Counter 0 and Counter 1 only)
CY_SYS_WDT_MODE_INT_RESET	Generates an interrupt and generates a reset on a 3rd unhandled interrupt. (valid for Counter 0 and Counter1 only)

Return Value: None.

Side Effects and Restrictions: WDT counter counterNum should be disabled to set a mode. Otherwise, this function call will have no effect. If the specified counter is enabled, call the CySysWdtDisable() function with the corresponding parameter to disable the specified counter and wait for it to stop.

uint32 CySysWdtGetMode(uint32 counterNum)

Description: Reads the mode of one of the three WDT counters.

Parameters: counterNum: The valid range [0-2]. Number of the WDT counter.

Return Value: The mode of the counter. The same enumerated values as mode parameters used in CySysWdtWriteMode().

void CySysWdtSetClearOnMatch(uint32 counterNum, uint32 enable)

Description: Configures the WDT counter clear on a match setting. If configured to clear on a match the counter counts from 0 to the MatchValue giving it a period of (MatchValue + 1).

Parameters: counterNum: Valid range [0-1]. Number of the WDT counter. Match values are not supported by Counter 2.

enable: 0 to disable, 1 to enable

Return Value: None.

Side Effects and Restrictions: WDT counter counterNum should be disabled. Otherwise, this function call will have no effect. If the specified counter is enabled, call the CySysWdtDisable() function with the corresponding parameter to disable the specified counter and wait for it to stop. This may take up to three LFCLK cycles.

uint32 CySysWdtGetClearOnMatch(uint32 counterNum)

Description: Reads the clear on a match setting for the specified counter.

Parameters: counterNum: The valid range [0-1]. Number of the WDT counter. Match values are not supported by Counter 2.

Return Value: Clear on Match status: 1 if enabled, 0 if disabled.

void CySysWdtSetCascade(uint32 cascadeMask)

Description: Writes the two WDT cascade values based on the combination of mask values specified.

Parameters: cascadeMask: Mask value used to set or clear both of the cascade values:

Define	Counter
CY_SYS_WDT_CASCADE_NONE	Neither
CY_SYS_WDT_CASCADE_01	Cascade 01
CY_SYS_WDT_CASCADE_12	Cascade 12

To set both cascade modes, two defines should be ORed:
(CY_SYS_WDT_CASCADE_01 | CY_SYS_WDT_CASCADE_12)

Return Value: None.

Side Effects and Restrictions: If only one cascade mask is specified, the second cascade is disabled. To set both cascade modes, two defines should be ORed:
(CY_SYS_WDT_CASCADE_01 | CY_SYS_WDT_CASCADE_12).

WDT counters that are part of the specified cascade should be disabled. Otherwise, this function call will have no effect. If the specified counter is enabled, call the CySysWdtDisable() function with the corresponding parameter to disable the specified counter and wait for it to stop. This may take up to 3 LFCLK cycles.

uint32 CySysWdtGetCascade(void)

Description: Reads the two WDT cascade values returning a mask of the bits set.

Parameters: None.

Return Value: The mask of a cascade values set:

Define	Cascade
CY_SYS_WDT_CASCADE_NONE	Neither
CY_SYS_WDT_CASCADE_01	Cascade 01
CY_SYS_WDT_CASCADE_12	Cascade 12

void CySysWdtSetMatch(uint32 counterNum, uint32 match)

Description: Configures the WDT counter match comparison value.

Parameters: counterNum: The valid range [0-1]. The number of the WDT counter. Match values are not supported by Counter 2.

match: The valid range [0-65535]. The value to be used to match against the counter.

Return Value: None

uint32 CySysWdtGetMatch(uint32 counterNum)

Description: Reads the WDT counter match comparison value.

Parameters: counterNum: The valid range [0-1]. The number of the WDT counter. Match values are not supported by Counter 2.

Return Value: A 16-bit match value.

void CySysWdtSetToggleBit(uint32 bits)

Description: Configures which bit in the WDT Counter 2 to monitor for a toggle. When that bit toggles, an interrupt is generated if the mode for Counter 2 has interrupts enabled.

Parameters: bit: The valid range [0-31]. Counter 2 bit to monitor for a toggle.

Return Value: None.

Side Effects and Restrictions: WDT Counter 2 should be disabled. Otherwise, this function call will have no effect.

If the specified counter is enabled, call the CySysWdtDisable() function with the corresponding parameter to disable the specified counter and wait for it to stop. This may take up to 3 LFCLK cycles.

uint32 CySysWdtGetToggleBit(void)

Description: Reads which bit in WDT Counter 2 is monitored for a toggle.

Parameters: None.

Return Value: The bit that is monitored (range of 0 to 31).

void CySysWdtLock(void)

Description:	Locks out configuration changes to the Watchdog timer registers and ILO configuration register.
Parameters:	None.
Return Value:	None.
Side Effects and Restrictions:	After this API is called, the ILO can't be disabled until calling of CySysWdtUnlock().

void CySysWdtUnlock(void)

Description:	Unlocks the Watchdog Timer configuration register.
Parameters:	None.
Return Value:	None.
Side Effects and Restrictions:	This API enables the ILO, if it was disabled.

void CySysWatchdogFeed(uint32 counterNum)

Description:	Feeds the Watchdog Counter before it causes the device reset. Supported only for WDT0 and WDT1 in Watchdog or Watchdog w/ Interrupts modes.
Parameters:	counterNum: The number of the WDT counter.
Return Value:	None.
Side Effects and Restrictions:	Clears the WDT Counter in Watchdog mode or clears the WDT interrupt in Watchdog w/ Interrupts mode. Does nothing in other modes.

uint32 CySysWdtGetCount(uint32 counterNum)

Description:	Reads the current WDT counter value.
Parameters:	counterNum: The valid range [0-2]. The number of the WDT counter.
Return Value:	A live counter value. Counter 0 and 1 are 16-bit counters and Counter 2 is a 32-bit counter.

void CySysWdtResetCounters(uint32 counterMask)

Description: Resets all WDT counters set in the mask.

Parameters: counterMask: The mask of all counters to reset:

Define	Counter
CY_SYS_WDT_COUNTER0_RESET	0
CY_SYS_WDT_COUNTER1_RESET	1
CY_SYS_WDT_COUNTER2_RESET	2

Return Value: None.

Side Effects and Restrictions: This function will not reset counter values if the Watchdog is locked.

uint32 CySysWdtGetInterruptSource(void)

Description: Reads a mask containing all the WDT interrupts that are currently set.

Parameters: None.

Return Value: The mask of interrupts set:

Define	Counter
CY_SYS_WDT_COUNTER0_INT	0
CY_SYS_WDT_COUNTER1_INT	1
CY_SYS_WDT_COUNTER2_INT	2

void CySysWdtClearInterrupt(uint32 counterMask)

Description: Clears all WDT counter interrupts set in the mask. Calling this API also prevents a reset from happening when Counter mode is set to generate 3 interrupts and then to reset the device. All WDT interrupts are to be cleared by the firmware, otherwise, interrupts are generated continuously.

Parameters: counterMask: The mask of all the counters to enable:

Define	Counter
CY_SYS_WDT_COUNTER0_INT	0
CY_SYS_WDT_COUNTER1_INT	1
CY_SYS_WDT_COUNTER2_INT	2

Return Value: None.

Side Effects and Restrictions: This function temporary removes the watchdog lock, if it was set, and restores the lock state, after cleaning the WDT interrupts, that are set in a mask.

void CySysSetInterruptCallback(uint32 counterNum, cyWdtCallback function)

Description:	Sets the ISR callback function for the particular WDT counter. These functions are called on the WDT interrupt.
Parameters:	counterNum : The number of the WDT counter. function : The pointer to the callback function.
Return Value:	The pointer to the previous callback function. NULL is returned if the specified address is not set.
Side Effects and Restrictions:	None.

void CySysGetInterruptCallback(uint32 counterNum, cyWdtCallback function)

Description:	Gets the ISR callback function for the particular WDT counter.
Parameters:	counterNum : The number of the WDT counter.
Return Value:	The pointer to the callback function registered for a particular WDT by a particular address that are passed through arguments.
Side Effects and Restrictions:	None.

void CySysWdtEnableCounterIsr(uint32 counterNum)

Description:	Enables the ISR callback servicing for the particular WDT counter.
Parameters:	counterNum: The valid range [0-2]. The number of the WDT counter.
Return Value:	None.
Side Effects and Restrictions:	None.

void CySysWdtDisableCounterIsr(uint32 counterNum)

Description:	Disables the ISR callback servicing for the particular WDT counter.
Parameters:	counterNum: The valid range [0-2]. The number of the WDT counter.
Return Value:	None.
Side Effects and Restrictions:	None.

void CySysWdtIsr(void)

Description:	This is the handler of the WDT interrupt in CPU NVIC. The handler checks which WDT triggered in the interrupt and calls the respective callback functions configured by the user by using CySysWdtSetIsrCallback() API. The order of the callback execution is incremental. Callback-0 is run as the first one and callback-2 is called as the last one.
Parameters:	None.
Return Value:	None.
Side Effects and Restrictions:	This function clears the WDT interrupt every time when it is called. A reset after the 3rd interrupt does not happen if this function is registered as the interrupt handler even if the Watchdog with Interrupt mode is selected on the **Low Frequency Clocks** tab.

void CySysTimerDelay(uint32 counterNum, cy_sys_timer_delaytype_enum delayType, uint32 delay)

Description: The function implements the delay specified in the LFCLK clock ticks. The specified WDT Counter should be configured as described below and started.

This function can operate in two modes: the WAIT and INTERRUPT modes. In the WAIT mode, the function waits for the specified number of ticks. In the INTERRUPT mode, the interrupt is generated after the specified number of ticks.

For the correct function operation, the Clear On Match" option should be disabled for the specified WDT counter. Use CySysWdtSetClearOnMatch() function with the “enable” parameter equals zero for the used WDT counter.

The corresponding WDT counter should be configured to match the selected mode: "Free running Timer" for the WAIT mode, and "Periodic Timer" / "Watchdog (w/Interrupt)" for the INTERRUPT mode.

This can be configured in two ways:

- Through the DWR page. Open the **Clocks** tab and click the **Edit Clocks…** button. In the “Configure System Clocks” window, click on the **Low Frequency Clocks** tab, and choose the appropriate option for the used WDT counter.
- Through the CySysWdtSetMode() function. Call it with the appropriate mode parameter for the used WDT counter.

For the INTERRUPT mode, the recommended sequence is the following:

- Call the CySysWdtDisableCounterIsr() function to disable servicing interrupts of the specified WDT counter.
- Call the CySysWdtSetInterruptCallback() function to register the callback function for the corresponding WDT counter.
- Call the CySysTimerDelay() function.

Parameters: counterNum: The valid range [0-1]. The number of the WDT Counter (WDT0 or WDT1).

delayType : Defines the operation mode.

- CY_SYS_TIMER_WAIT – WAIT mode.
- CY_SYS_TIMER_INTERRUPT – INTERRUPT mode.

delay: The delay value in the LFCLK ticks (allowable range - 16-bit value).

Return Value: None.

Side Effects and Restrictions: In INTERRUPT mode, this function enables ISR callback servicing from the corresponding WDT counter. Servicing of this ISR callback is disabled after the expiration of the delay time.

void CySysTimerDelayUntilMatch(uint32 counterNum, cy_sys_timer_delaytype_enum delayType, uint32 delay)

Description: The function implements a delay specified as the number of LFCLK ticks between the specified WDT counter's current value and the "match" passed as the parameter to this function. The current WDT counter value can be obtained using the CySysWdtGetCount() function.

This function can operate in two modes: WAIT and INTERRUPT. In WAIT mode, the function waits while the used WDT counter value becomes equal to the match value. In INTERRUPT mode, the interrupt is generated when the used WDT counter becomes equal to the "match" value.

For correct function operation, the Clear On Match option should be disabled for the specified WDT counter. Use the CySysWdtSetClearOnMatch() function with the "enable" parameter that equals zero for the used WDT counter.

The corresponding WDT counter should be configured to match the selected mode: "Free running Timer" for WAIT mode, and "Periodic Timer" / "Watchdog (w/Interrupt)" for INTERRUPT mode.

This can be configured in two ways:

- Through the DWR page. Open the **Clocks** tab and click the **Edit Clocks…** button. In the Configure System Clocks window, click on the Low **Frequency Clocks** tab, and choose the appropriate option for the used WDT counter.
- Through the CySysWdtSetMode() function. Call it with the appropriate mode parameter for the used WDT counter.

For INTERRUPT mode, the recommended sequence is the following:

- Call the CySysWdtDisableCounterIsr() function to disable servicing interrupts of the specified WDT counter.
- Call the CySysWdtSetInterruptCallback() function to register the callback function for the corresponding WDT counter.
- Call the CySysTimerDelay() function.

Parameters: counterNum: The valid range [0-1]. The number of the WDT counter (WDT0 or WDT1).

delayType : Defines the operation mode.

- CY_SYS_TIMER_WAIT – WAIT mode.
- CY_SYS_TIMER_INTERRUPT – INTERRUPT mode.

delay: The delay value in the LFCLK ticks (allowable range - 16-bit value).

Return Value: None.

Side Effects and Restrictions: In INTERRUPT mode, this function enables ISR callback servicing from the corresponding WDT counter. Servicing of this ISR callback is disabled after the expiration of the delay time.

WDT configuration (PSoC 4000)

Function	Description
CySysWdtEnable	Enables the watchdog timer reset generation. CySysWdtClearInterrupt() feeds the watchdog. Two unserviced interrupts lead to a system reset (that is, at the third match).
CySysWdtDisable	Disables the WDT reset generation.
CySysWdtGetEnabledStatus	Reads the enabled status of the WDT counter.
CySysWdtSetMatch	Configures the WDT counter match comparison value.
CySysWdtGetMatch	Reads the WDT counter match comparison value.
CySysWdtSetIgnoreBits	Configures the number of the MSB bits of the watchdog timer that are not checked against the match.
CySysWdtGetIgnoreBits	Reads the number of the MSB bits of the watchdog timer that are not checked against the match.
CySysWdtClearInterrupt	Feeds the watchdog. Cleans the WDT match flag which is set every time the WDT counter reaches a WDT match value. Two unserviced interrupts lead to a system reset (i.e. at the third match).
CySysWdtMaskInterrupt	After masking interrupts from WDT, they are not passed to CPU. This function does not disable WDT reset generation.
CySysWdtUnmaskInterrupt	After unmasking interrupts from WDT, they are passed to CPU. This function does not impact the reset generation.
CySysWdtGetCount	Reads the current WDT counter value.
CySysWdtSetIsrCallback	Sets the ISR callback function for the particular WDT counter.
CySysWdtGetIsrCallback	Gets the ISR callback function for the particular WDT counter.
CySysWdtIsr	This is the handler of the WDT interrupt in CPU NVIC. The handler calls the respective callback functions configured by the user by using CySysWdtSetIsrCallback() API.

void CySysWdtEnable(void)

Description: Enables the watchdog timer reset generation. The CySysWdtClearInterrupt() feeds the Watchdog. Two unserviced interrupts lead to a system reset (i.e. at the third match).

Parameters: None.

Return Value: None.

Side Effects and Restrictions: The ILO is enabled by the hardware once WDT is started.

void CySysWdtDisable(void)

Description: Disables the WDT reset generation.

Parameters: None.

Return Value: None.

uint32 CySysWdtGetEnabledStatus(void)

Description: Reads the enabled status of the WDT Counter.

Parameters: None.

Return Value: The status of the WDT counter:
0 - counter is disabled
1 - counter is enabled

void CySysWdtSetMatch(uint32 match)

Description: Configures the WDT counter match comparison value.

Parameters: match: The valid range [0-65535]. The value to be used to match against the counter.

Return Value: None.

uint32 CySysWdtGetMatch(void)

Description: Reads the WDT counter match comparison value.

Parameters: None.

Return Value: The counter match value.

void CySysWdtSetIgnoreBits(uint32 bitsNum)

Description: Configures the number of MSB bits of the Watchdog timer that are not checked against a match.

Parameters: bitsNum: The valid range [0-15]. The number of MSB bits.

Return Value: None.

Side Effects and Restrictions: The value of bitsNum that provides control over the time-to-reset of the Watchdog (which happens after 3 successive matches).

uint32 CySysWdtGetIgnoreBits(void)

Description: Reads the number of MSB bits of the Watchdog timer that are not checked against a match.

Parameters: None.

Return Value: The number of MSB bits.

void CySysWdtClearInterrupt(void)

Description: Feeds the Watchdog. Cleans the WDT match flag, which is set every time the WDT counter reaches a WDT match value. Two unserviced interrupts lead to a system reset (i.e. at the third match).

Parameters: None.

Return Value: None.

void CySysWdtMaskInterrupt(void)

Description: After masking interrupts from WDT, they are not passed to CPU. This function does not disable the WDT reset generation on 2 missed interrupts.

Parameters: None.

Return Value: None.

void CySysWdtUnmaskInterrupt(void)

Description: Unmasks WDT interrupts.

Parameters: None.

Return Value: None.

uint32 CySysWdtGetCount(void)

Description: Reads the current WDT counter value.

Parameters: None.

Return Value: A live counter value.

uint32 CySysWdtSetInterruptCallback(cyWdtCallback function)

Description: Sets the ISR callback function for the particular WDT Counter.

Parameters: function: The pointer to the callback function.

Return Value: The pointer to a previous callback function.

uint32 CySysWdtGetInterruptCallback(void)

Description: Gets the ISR callback function for the particular WDT counter.

Parameters: None.

Return Value: The pointer to a previous callback function.

void CySysWdtIsr(void)

Description: This is the handler of the WDT interrupt in CPU NVIC. The handler calls the respective callback functions configured by the user by using CySysWdtSetIsrCallback() API.

Parameters: None.

Return Value: None.

Side Effects and Restrictions: This function clears the WDT interrupt every time when it is called. A reset after the 3rd interrupt does not happen if this function is registered as the interrupt handler even if the Watchdog with Interrupt mode is selected on the **Low Frequency Clocks** tab.

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.

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.

API Memory Usage

Please refer to the cy_boot System Reference Guide.

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 The data for the PSoC 4200L device is preliminary. Final data will be delivered in an upcoming Component Pack.

AC Specifications

Parameter	Description	Min	Typ	Max	Units	Conditions
FILOTRIM1	32 kHz trimmed frequency	15	32	50	kHz	±60% with trim.
FILOTRIM1	32 kHz trimmed frequency for PSoC 4000	20	40	80	kHz	
TSTARTILO1	ILO startup time	-	-	2	ms	
F_{WCO}	Crystal Frequency		32.768		kHz	
F_{TOL}	Frequency tolerance		50	250	ppm	With 20 ppm crystal.
E_{SR}	Equivalent series resistance		50		kΩ	
T_{START}	Startup time			500	ms	WCO settling delay during System boot
C_L	Crystal Load Capacitance	6		12.5	pF	
C_0	Crystal Shunt Capacitance		1.35		pF	

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.0.c	Updated datasheet.	Added note that the data for the PSoC 4200L device is preliminary.
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.

Version	Description of Changes	Reason for Changes / Impact
		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, 2016. 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.