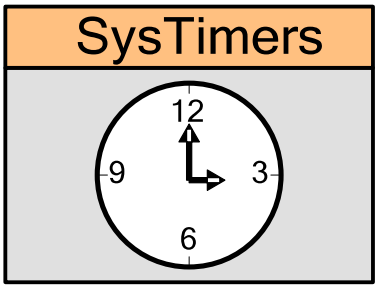
SysTimers
3.0

Features



- Uses PSoC 4/5LP SysTick Timer
- Requires no internal hardware or GPIO pins.
- Up to 16 parallel timers
- Timer updates in a single ISR
- Nine timer resolutions between 25uS and 250mS
- Two modes of operation
- Callback functions may be assigned to timers

General Description

The SysTimers component makes use of the Cortex M0/M3 SysTick timer to create 2 to 16 non-blocking timers. The SysTick interrupt period is set by the component and increments the timer/s at each interrupt. These timers provide a way to time or schedule parallel periodic events without consuming valuable hardware or making use of blocking functions such as CyDelay().

When to Use a SysTimers Component

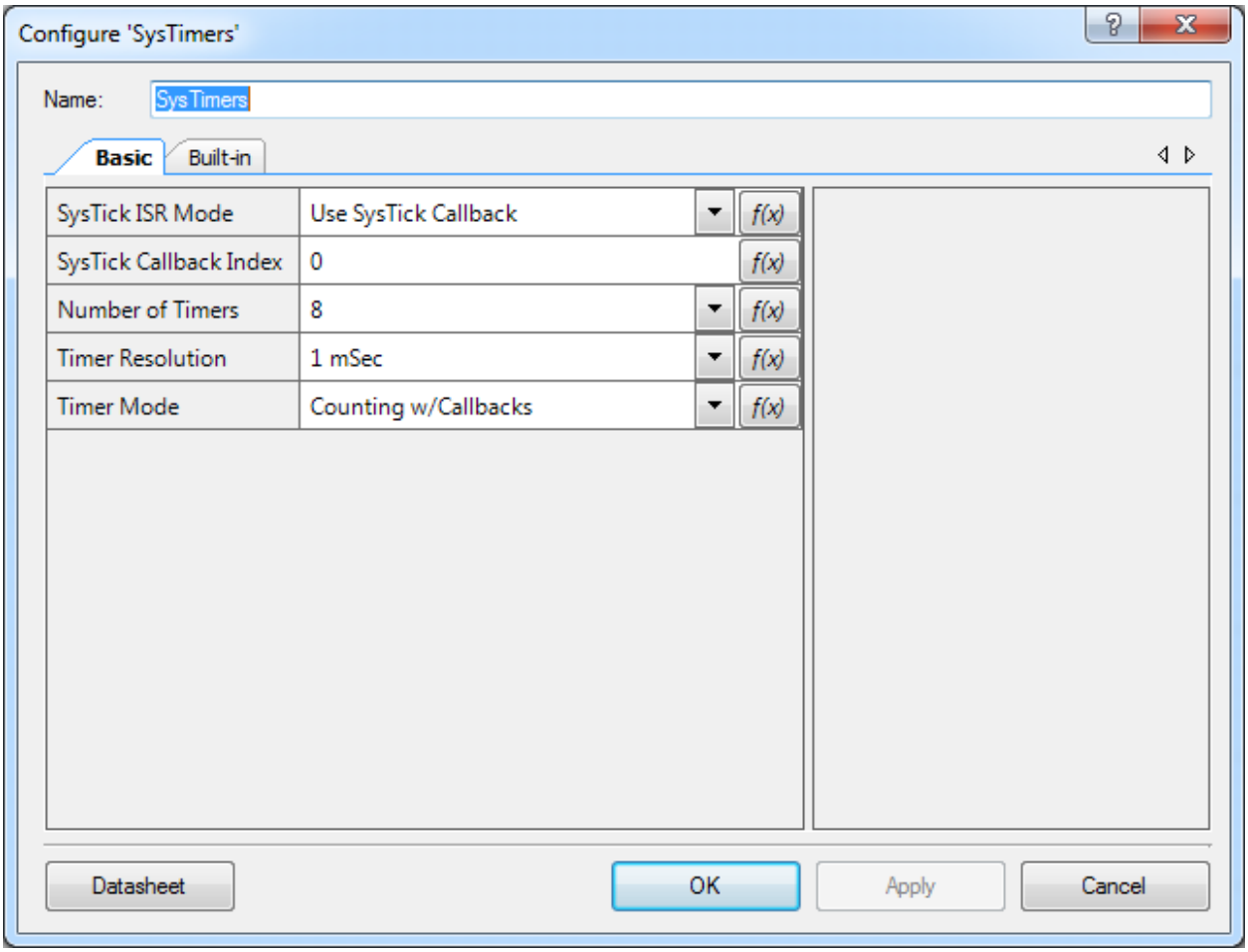
Use the SysTimers component when relatively slow events or delays need to be timed, but without using blocking code such as with CyDelay() .

Input/Output Connections

There are no Input or Output pins on this component.

Component Parameters

Drag a SysTimers component onto your design and double-click it to open the Configure dialog.



Parameters

SysTick ISR Mode

This parameter lets you either replace the Systick ISR with the System Timers ISR or to use the default system ISR and post a callback function. If the System Timer is the only part of the design that uses SysTick, then choose the “Replace Systick ISR” option for more efficient code. If SysTick is used for more than just the System Timers, you will need to choose the “Use SysTick Callback” option which allows up to four callback functions to be called by the SysTick ISR. Also, if you select “Use SysTick Callback” mode, make sure you turn on the system SysTick timer with the system command, `CySysTickStart()`. When using the “Replace SysTick ISR” mode, the SysTick timer is automatically started when you start the timers with the `SysTimers_Start()` function.

SysTick Callback Index

This parameter is only visible if “Use SysTick Callback” is selected for the ISR Mode parameter. It sets the index parameter for the CySysTickSetCallback() function, which will be automatically done when calling the _Start() function. This value should be set between 0 and 3 with a default value of 0. PSoC Creator by default allows up to 4 functions to automatically be called at the end of each SysTick period (when SysTick timer increments). This index defines which of the entries will be used for the SystemTimers. If the “Replace SysTick ISR” mode is used, this parameter is invalid.

Number of Timers

This parameter allows you to select the maximum timers that you require in your design. The options are 2, 4, 8, and 16, with 4 being the default.

Timer Mode

There are two timer modes “Counting w/Callbacks” and “Fast IRQ”. The Fast IRQ has very little overhead in the SysTick ISR, since it only increments a counter. This is the best option if your application does not require a callback functions associated with the timer, or need to know if the timer period has lapsed more than once.

If using the System Timer Callback functions, the “Counting w/Callbacks” option needs to be selected. During each SysTick interrupt, each timer is checked for status and calls the timer callback function if one has been assigned.

- Fast IRQ
- Counting w/Callbacks

Timer Resolution

This parameter sets the timer resolution. It is basically how often the SysTick ISR is invoked to update the counter/s. The default period is 1 mSec.

- 250 mSec
- 100 mSec
- 25 mSec
- 10 mSec
- 2.5 mSec
- **1 mSec**

- 250 uSec
- 100 uSec
- 50 uSec
- 25 uSec

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 together with related constants provided by the "include" files. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "SysTimers_1" to the first instance of a component in a given project. 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 "SysTimers."

Functions	Description
SysTimers_Start()	Initializes timers, sets the SysTick period and enables the interrupt.
SysTimers_Stop()	Disables timers by disabling the interrupt.
SysTimers_OverRideSysPeriod()	Overrides the system SysTick Period .
SysTimers_GetTimer()	Get a free timer and set the period
SysTimers_GetTimerCB()	Get a free timer, set the period and the CallBack function pointer.
SysTimers_SetCallback()	Sets a callback function for an existing timer.
SysTimers_GetTimerStatus()	Check to see if timer has expired
SysTimers_GetTimerValue()	Get timer value
SysTimers_GetSysTickValue()	Return value of SysTick timer.
SysTimers_ResetTimer()	Reset the period of a specific timer.
SysTimers_ReleaseTimer()	Release a timer for use so that it can be used by another part of the firmware.
SysTimers_ReleaseAllTimers()	Release all timers and reset their period back to default.
SysTimers_ResumeTimer()	Resume a suspended timer.
SysTimers_SuspendTimers()	Suspend a timer.

void SysTimers_Start(void)

Description: This function sets the period of the SysTick interrupt to the selected value, initializes the timers, and enables the SysTick interrupt.

Parameters: None

Return Value: None

Side Effects: None

If the **ISR Mode** is set to "**Use SysTick Callback**", it is important that the system call CySysTickStart() be called before calling the SysTimers_Start() function. Since the "**Use SysTick Callback"** mode uses the default system period, you may call SysTimers_OverRideSysPeriod() to get the timer resolution selected in the interface instead of relying on the default setting. Also, if the ISR Mode is set to "Replace SysTickISR" there is no need to call CySysTickStart() or SysTimers_OverRideSysPeriod(). Below is a code snippit handy to use if experimenting with both modes.

```
#if (SysTimers_ISR_MODE == SysTimers_ISR_MODE_CALLBACK)
      CySysTickStart();
      SysTimers_OverRideSysPeriod();
#endif

SysTimers_Start();
```

void SysTimers_Stop(void)

Description: Disables the SysTick interrupt so that all timers will not update.

Parameters: None

Return Value: None

Side Effects: None

void SysTimers_OverRideSysPeriod(void)

Description: This function overrides the default system SysTick Period. It will set the SysTick Period to the resolution selected in the SysTimers user interface. This function is not require is the ISR_Mode is set to "Replace SysTick ISR".

Parameters: None

Return Value: None

Side Effects: None

uint32 SysTimers_GetTimer(uint32 timerPeriod)

Description: Returns a timer ID from the timer set to the requested period.

Parameters: uint32 timerPeriod: The period in SysTick interrupt periods For example, if the timer resolution is set to 1mSec and the timer Value is 100, the timer will expire in 100 mSec or at a 10 Hz rate.

Return Value: uint32: The timer ID of the requested timer. Use this ID when calling the functions GetTimerStatus(), GetTimerValue(), ResetTimer(), or ReleaseTimer(). If all timers are in use, a zero will be returned.

Side Effects: None

uint32 SysTimers_GetTimerCB(uint32 timerPeriod, SysTimers_Callback_t function)

Description: Returns a timer ID from the timer set to the requested period. Must set "Timer Mode" to "Counting w/Callback" to use this function.

Parameters: uint32 timerPeriod: The period in SysTick interrupt periods For example, if the timer resolution is set to 1mSec and the timer Value is 100, the timer will expire in 100 mSec or at a 10 Hz rate.

SysTimers_Callback_t function: Pointer to the callback function. This function will be called upon expiration of the timer.

Return Value: uint32: The timer ID of the requested timer. Use this ID when calling the functions GetTimerStatus(), GetTimerValue(), ResetTimer(), or ReleaseTimer(). If all timers are in use, a zero will be returned.

Side Effects: None

uint32 SysTimers_SetCallback(uint32 index, SysTimers_Callback_t function)

Description: Returns a timer ID from the timer set to the requested period. Must set "Timer Mode" to "Counting w/Callback" to use this function.

Parameters: uint32 index: Timer index in which to set the callback function.

SysTimers_Callback_t function: Pointer to the callback function. This function will be called upon expiration of the timer. Pass a null function pointer to disable the callback function.

Return Value: uint32: The timer ID of the requested timer. Use this ID when calling the functions GetTimerStatus(), GetTimerValue(), ResetTimer(), or ReleaseTimer(). If all timers are in use, SysTimers_INVALID_TIMER will be returned.

Side Effects: None

uint32 SysTimers_GetTimerStatus(uint32 timerID)

Description: Returns the status for timer specified by timerID.

Parameters: uint32 timerID: Timer ID for the specific timer, returned from GetTimer().

Return Value: uint32: If timer has expired, a non-zero value will be returned, otherwise a zero will be returned. If operating in the “Counting” mode, the return value will be the count of how many of the set periods has passed since the last time this function was called.

Side Effects: The status is automatically cleared when this function is called.

uint32 SysTimers_GetTimerValue(uint32 timerID)

Description: Returns the counter value for the specific timer specified by timerID .

Parameters: uint32 timerID: Timer ID for the specific timer, returned from GetTimer().

Return Value: Returns how many SysTicks until the counter expires in the Counting mode and the amount of SysTicks since the timer started in the FastIRQ mode. If timer

Side Effects: None

uint32 SysTimers_GetSysTickValue(void)

Description: Returns the value of the SysTick counter or “SysTimers_SysTickCount”. This counter was set to zero when the component was started. It is a 32-bit counter so if the resolution is set to 1mSec, the timer will roll over in about 49 days.

Parameters: None

Return Value: uint32: Value of the SysTick counter, “SysTimers_SysTickCount”.

Side Effects: None

uint32 SysTimers_ResetTimer(uint32 timerID, uint32 timerPeriod)

Description: Resets the timer with a new or original period, starting at the call of this function.

Parameters: uint32 timerID: ID of the timer that will be reset.
uint32 timerPeriod: New timer period. If this value is zero, the old period will be reloaded.

Return Value: None

Side Effects: None

uint32 SysTimers_ReleaseTimer(uint32 timerID)

Description: Releases the specified timer so it can be used elsewhere.

Parameters: uint32 timerID: ID of the specific timer to release.

Return Value: None

Side Effects: None

void SysTimers_ReleaseAllTimers(void)

Description: Releases all timers. User must use GetTimer() function to resume using any timers.

Parameters: None.

Return Value: None

Side Effects: All timer IDs will become invalid.

uint32 SysTimers_ResumeTimer(uint32 timerID)

Description: Resumes a once suspended timer.

Parameters: uint32 timerID: ID of the specific timer to resume.

Return Value: None

Side Effects: None

uint32 SysTimers_SuspendTimer(uint32 timerID)

Description: Suspends a timer until resumed or released.

Parameters: uint32 timerID: ID of the specific timer to suspend.

Return Value: None

Side Effects: None

Resources

The SysTick interrupt is used by the SysTimers component.

API Memory Usage

SRAM usage is 12 bytes per channel plus 4 bytes overhead. For typical usage, Flash will be less than 500 bytes. Flash usage is not dependent on the amount of channels selected.

DC and AC Electrical Characteristics

N/A

Component Changes

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

Version	Description of Changes
1.0	Initial Release
1.1	Replaced “device.h” with “project.h” in SysTimers.c Arranged timing options from longest to shortest. Added description of SysTimers_GetTimerStatus () Added more timer resolutions
2.0	Added option not to override the System SysTick ISR and use a Callback function instead. Added SysTimers_OverRideSysPeriod() function.
2.0a	Updated the customizer to hide invalid parameters with PSoC Creator 4.2 features. No change in functionality.
3.0	Added callback functions GetTimerCB(), SetCallback() and ResumeTimer()/SuspendTimer().