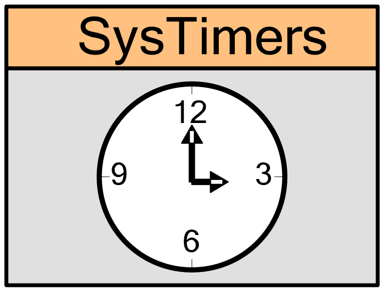
SysTimers
1.1



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

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

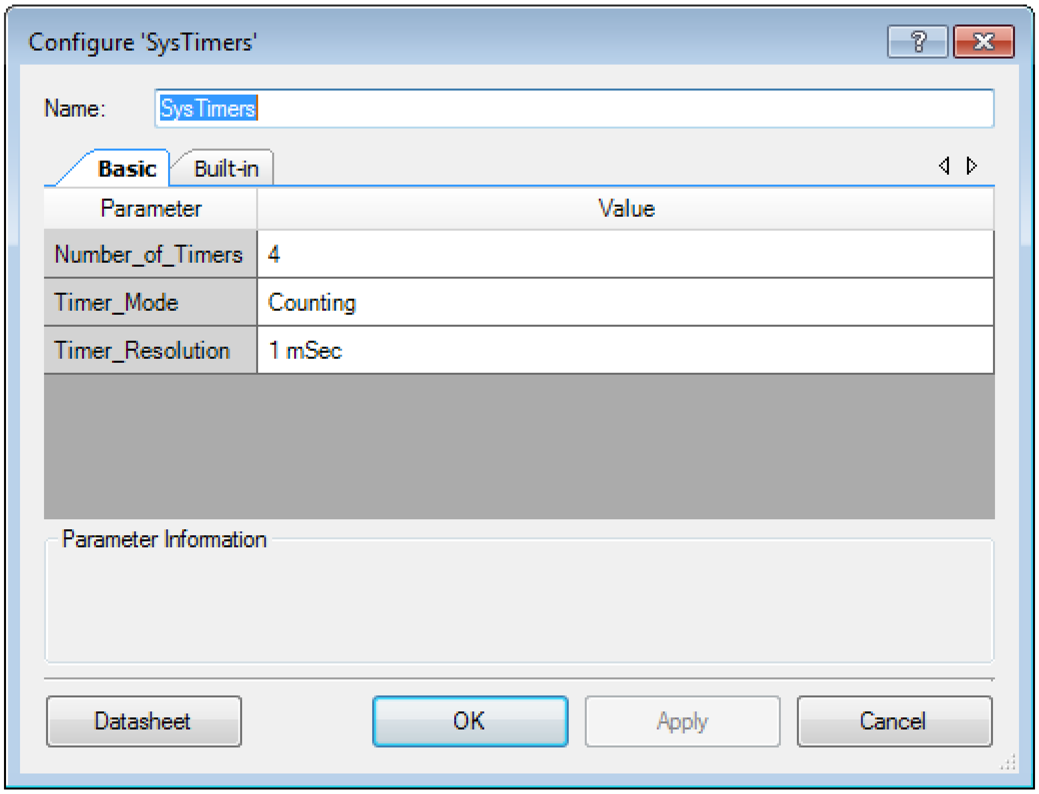
Use the SysTimers component when relatively slow events or delays need to be timed, but without using blocking code such as with CyDelay() .

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.



Parameter	Value
Number_of_Timers	4
Timer_Mode	Counting
Timer_Resolution	1 mSec

Parameters

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.

Time Mode

There are two modes “Counting” and “FastIRQ”. For most applications, they are identical, except the FastIRQ has minimal code in the ISR. The Counting mode returns the number of periods that have expired since the timer was started or the last time the GetTimerStatus() function was called. The FastIRQ mode will just return a non-zero if the timer period has expired.

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
- 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_GetTimer()	Get a free timer and set the period
SysTimers_GetTimerStatus()	Check to see if timer has expired
SysTimers_GetTimerValue()	Get remaining timer period
SysTimers_GetSysTickValue()	Return value of SysTick timer.
SysTimers_ResetTimer()	Reset the period of a specific timer.

Functions	Description
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.

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

void SysTimers_Stop(void)

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

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

void 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

void 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.

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