



## Quality RTOS & Embedded Software

[About](#) [Contact](#) [Support](#) [FAQ](#)



[Quick Start](#) [Supported MCUs](#) [Books & Kits](#) [Trace Tools](#) [Ecosystem](#) [TCP & FAT](#) [Training](#) [Email List+](#) [Twitter](#) [RSS](#)

### Home

FreeRTOS Books and Manuals

#### FreeRTOS

[About FreeRTOS](#)

[Features / Getting Started...](#)

[More Advanced...](#)

[Demo Projects](#)

[Supported Devices & Demos](#)

[API Reference](#)

[PDF Reference Manual](#)

[Task Creation](#)

[Task Control](#)

[Task Utilities](#)

[RTOS Kernel Control](#)

[Direct To Task Notifications](#)

[FreeRTOS-MPU Specific](#)

[Queues](#)

[Queue Sets](#)

[Semaphore / Mutexes](#)

[Software Timers](#)

[xTimerCreate\(\)](#)

[xTimerIsTimerActive\(\)](#)

[xTimerStart\(\)](#)

[xTimerStop\(\)](#)

[xTimerChangePeriod\(\)](#)

[xTimerDelete\(\)](#)

[xTimerReset\(\)](#)

[xTimerStartFromISR\(\)](#)

[xTimerStopFromISR\(\)](#)

[xTimerChangePeriodFromISR\(\)](#)

[xTimerResetFromISR\(\)](#)

[pvTimerGetTimerID\(\)](#)

[vTimerSetTimerID\(\)](#)

[xTimerGetTimerDaemonTaskHandle\(\)](#)

[xTimerPendFunctionCall\(\)](#)

[xTimerPendFunctionCallFromISR\(\)](#)

[Event Groups \(or 'flags'\)](#)

[Co-routines](#)

[Contact, Support, Advertising](#)

[FreeRTOS Interactive!](#)

[Quick Start Guide](#)

[Download Source](#)

### FreeRTOS+ Lab Projects

#### FreeRTOS+TCP:

Thread safe TCP/IP stack

#### FreeRTOS+FAT:

Thread aware file system

### FreeRTOS+ Ecosystem

#### Internet of Things:

Innovative complete solution

#### Fail Safe File System:

Ensures data integrity

#### InterNiche TCP/IP:

Low cost pre-ported libraries

#### FreeRTOS BSPs:

3<sup>rd</sup> party driver packages

#### FAT SL File System:

Super lean FAT FS

#### UDP/IP:

Thread aware UDP stack

#### Trace & Visualisation:

Tracealyzer for FreeRTOS

## xTimerStart

### [Timer API]

timers.h

```
BaseType_t xTimerStart( TimerHandle_t xTimer,
                        TickType_t xBlockTime );
```

Timer functionality is provided by a timer service/daemon task.

Many of the public FreeRTOS timer API functions send commands to the timer service task through a queue called the timer command queue. The timer command queue is private to the RTOS kernel itself and is not directly accessible to application code. The length of the timer command queue is set by the configTIMER\_QUEUE\_LENGTH configuration constant.

xTimerStart() starts a timer that was previously created using the [xTimerCreate\(\)](#) API function. If the timer had already been started and was already in the active state, then xTimerStart() has equivalent functionality to the [xTimerReset\(\)](#) API function.

Starting a timer ensures the timer is in the active state. If the timer is not stopped, deleted, or reset in the mean time, the callback function associated with the timer will get called 'n' ticks after xTimerStart() was called, where 'n' is the timers defined period.

It is valid to call xTimerStart() before the RTOS scheduler has been started, but when this is done the timer will not actually start until the RTOS scheduler is started, and the timers expiry time will be relative to when the RTOS scheduler is started, not relative to when xTimerStart() was called.

The configUSE\_TIMERS configuration constant must be set to 1 for xTimerStart() to be available.

### Parameters:

<i>xTimer</i>	The handle of the timer being started/restarted.
<i>xBlockTime</i>	Specifies the time, in ticks, that the calling task should be held in the Blocked state to wait for the start command to be successfully sent to the timer command queue, should the queue already be full when xTimerStart() was called. xBlockTime is ignored if xTimerStart() is called before the RTOS scheduler is started.

### Returns:

pdFAIL will be returned if the start command could not be sent to the timer command queue even after xBlockTime ticks had passed. pdPASS will be returned if the command was successfully sent to the timer command queue. When the command is actually processed will depend on the priority of the timer service/daemon task relative to other tasks in the system, although the timers expiry time is relative to when xTimerStart() is actually called. The timer service/daemon task priority is set by the configTIMER\_TASK\_PRIORITY configuration constant.

### Example usage:

### Latest News:

FreeRTOS V9.0.0rc1 is now available for [download](#) and [comment](#).

### Buildable Examples

FreeRTOS+TCP



FreeRTOS+FAT

[Try Them Now](#)

### Sponsored Links

[Now With No Code Size Limit!](#)



**CLI:**

Command line interface

**WolfSSL SSL / TLS:**

Networking security protocols

**Safety:**

TUV certified RTOS

**RTOS Training:**

Delivered online or on-site

**IO:**

read(), write(), ioctl() interface

See the example on the [xTimerCreate\(\)](#) documentation page.

[\[ Back to the top \]](#)   [\[ About FreeRTOS \]](#)   [\[ Sitemap \]](#)   [\[ Report an error on this page \]](#)

Google™ Custom Search

Search

Copyright (C) 2004-2010 Richard Barry. Copyright (C) 2010-2016 Real Time Engineers Ltd.  
Any and all data, files, source code, html content and documentation included in the FreeRTOS™ distribution or available on this site are the exclusive property of Real Time Engineers Ltd.. See the files license.txt (included in the distribution) and this [copyright notice](#) for more information. FreeRTOS™ and FreeRTOS.org™ are trade marks of Real Time Engineers Ltd.

Atmel®



XILINX®

ALTERA®

freescale™  
Alliance Member

infineon

TEXAS  
INSTRUMENTS  
MCU  
Developer Network

FUJITSU

Microsemi

a atollic

IAR  
SYSTEMSKEIL™  
Tools by ARMEmbedded  
Artists