



Quality RTOS & Embedded Software
[About](#) [Contact](#) [Support](#) [FAQ](#)

FreeRTOS Tutorial Books and Reference Manuals
Become an expert, while supporting the FreeRTOS project

[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
 xSemaphoreCreateBinary()
 vSemaphoreCreateBinary()
 xSemaphoreCreateCounting()
 xSemaphoreCreateMutex()
 xSemaphoreCreateRecursiveMutex()
 vSemaphoreDelete()
 xSemaphoreGetMutexHolder()
 xSemaphoreTake()
 xSemaphoreTakeFromISR()
 xSemaphoreTakeRecursive()
 xSemaphoreGive()
 xSemaphoreGiveRecursive()
 xSemaphoreGiveFromISR()
 Software Timers
 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:
 3rd party driver packages
 FAT SL File System:
 Super lean FAT FS
 UDP/IP:
 Thread aware UDP stack
 Trace & Visualisation:
 Tracealyzer for FreeRTOS
 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

xSemaphoreCreateMutex

[Semaphores]

Only available from FreeRTOS V4.5.0 onwards.

semphr. h

```
SemaphoreHandle_t xSemaphoreCreateMutex( void )
```

Macro that creates a mutex semaphore by using the existing queue mechanism.

Mutexes created using this macro can be accessed using the xSemaphoreTake() and xSemaphoreGive() macros. The xSemaphoreTakeRecursive() and xSemaphoreGiveRecursive() macros should not be used.

Mutexes and [binary semaphores](#) are very similar but have some subtle differences: Mutexes include a priority inheritance mechanism, binary semaphores do not. This makes binary semaphores the better choice for implementing synchronisation (between tasks or between tasks and an interrupt), and mutexes the better choice for implementing simple mutual exclusion.

The priority of a task that 'takes' a mutex can potentially be raised if another task of higher priority attempts to obtain the same mutex. The task that owns the mutex 'inherits' the priority of the task attempting to 'take' the same mutex. This means the mutex must always be 'given' back - otherwise the higher priority task will never be able to obtain the mutex, and the lower priority task will never 'disinherit' the priority. An example of a mutex being used to implement mutual exclusion is provided on the [xSemaphoreTake\(\)](#) documentation page.

A binary semaphore need not be given back once obtained, so task synchronisation can be implemented by one task/interrupt continuously 'giving' the semaphore while another continuously 'takes' the semaphore. This is demonstrated by the sample code on the [xSemaphoreGiveFromISR\(\)](#) documentation page.

Both mutex and binary semaphores are assigned to variables of type SemaphoreHandle_t and can be used in any API function that takes a parameter of this type.

Return:

Handle to the created semaphore. Should be of type SemaphoreHandle_t.

Example usage:

```
SemaphoreHandle_t xSemaphore;  
  
void vATask( void * pvParameters )  
{  
    // Mutex semaphores cannot be used before a call to  
    // xSemaphoreCreateMutex(). The created mutex is returned.  
    xSemaphore = xSemaphoreCreateMutex();  
  
    if( xSemaphore != NULL )  
    {  
        // The semaphore was created successfully.  
        // The semaphore can now be used.  
    }  
}
```

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! ↓



The best **UNLIMITED FREE** ARM[®] development on the planet.

NO CODE-SIZE LIMITATION


↑ Free Download Without Registering ↑



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.

