FreeRTOS v10.2.0 task.c 中相关的函数

任务创建API

xTaskCreate

```
BaseType_t xTaskCreate(TaskFunction_t pvTaskCode,
                       const char * const pcName,
                       configSTACK_DEPTH_TYPE usStackDepth,
                       void *pvParameters,
                       UBaseType_t uxPriority,
                       TaskHandle_t *pvCreatedTask);
```

创建一个任务，并且自动为其分配**栈空间**和**数据空间**。

xTaskCreateStatic

```
TaskHandle_t xTaskCreateStatic(TaskFunction_t pvTaskCode,
                               const char * const pcName,
                               uint32_t ulStackDepth,
                               void *pvParameters,
                               UBaseType_t uxPriority,
                               StackType_t *pxStackBuffer,
                               StaticTask_t *pxTaskBuffer );
```

创建一个任务，此时需要程序员来提供一个**栈空间**和**数据空间**的地址。

xTaskCreateRestricted

```
BaseType_t xTaskCreateRestricted(TaskParameters_t *pxTaskDefinition,
TaskHandle_t *pxCreatedTask );
```

创建一个任务，此时需要程序员手动提供一个**栈空间**地址。

xTaskCreateRestrictedStatic

```
 BaseType_t xTaskCreateRestrictedStatic(
            TaskParameters_t *pxTaskDefinition,
            TaskHandle_t *pxCreatedTask );
```

创建一个任务，需要程序员手动提供**栈空间**地址。**数据空间**会被自动地动态分配。

vTaskAllocateMPURegions

```
void vTaskAllocateMPURegions( TaskHandle_t xTask,
                              const MemoryRegion_t * const pxRegions );
```

用来为 `restricted task` 分配内存空间。

vTaskDelete

```
1   void vTaskDelete( TaskHandle_t xTask );
```

删除一个任务。

任务控制API

vTaskDelay

```
1   void vTaskDelay( const TickType_t xTicksToDelay );
```

以给定参数来延迟任务。

vTaskDelayUntil

```
1   void vTaskDelayUntil( TickType_t *pxPreviousWakeTime,
2                         const TickType_t xTimeIncrement );
```

指定某个确定的时间点来解除阻塞。

xTaskAbortDelay

```
1   BaseType_t xTaskAbortDelay( TaskHandle_t xTask );
```

让任务从跳出阻塞状态回到它原来被调用的地方。

uxTaskPriorityGet

```
1   UBaseType_t uxTaskPriorityGet( const TaskHandle_t xTask );
```

获得任务的优先级。

eTaskGetState

```
1   eTaskState eTaskGetState( TaskHandle_t xTask );
```

获取任务的状态码，是一个枚举类型。

vTaskGetInfo

```
1   void vTaskGetInfo( TaskHandle_t xTask, TaskStatus_t *pxTaskStatus,
    BaseType_t xGetFreeStackSpace, eTaskState eState );
```

获取任务的信息。

vTaskPrioritySet

```
1    void vTaskPrioritySet( TaskHandle_t xTask, UBaseType_t uxNewPriority );
```

设置任务的优先级。

vTaskSuspend

```
1    void vTaskSuspend( TaskHandle_t xTaskToSuspend );
```

挂起任务。

vTaskResume

```
1    void vTaskResume( TaskHandle_t xTaskToResume );
```

继续执行被挂起的任务。

程序调度

vTaskStartScheduler

```
1    void vTaskStartScheduler( void );
```

启动任务调度程序。

vTaskEndScheduler

```
1    void vTaskEndScheduler( void );
```

停止任务调度程序，在处理完后，又重新从 `vTaskStartScheduler` 开始。

vTaskSuspendAll

```
1    void vTaskSuspendAll( void );
```

在不终止中断的情况下挂起调度程序。

xTaskResumeAll

```
1    BaseType_t xTaskResumeAll( void );
```

继续执行被挂起的调度程序。

xTaskGetTickCount

```
1    TickType_t xTaskGetTickCount( void );
```

获取从 `vTaskStartScheduler` 被调用到现在的毫秒数。

uxTaskGetNumberOfTasks

```
uint16_t uxTaskGetNumberOfTasks( void );
```

返回内核正在管理的任务的子总数目。

pcTaskGetName

```
char *pcTaskGetName( TaskHandle_t xTaskToQuery );
```

返回任务的名字。

xTaskGetHandle

```
TaskHandle_t xTaskGetHandle( const char *pcNameToQuery );
```

返回任务的句柄。

uxTaskGetStackHighWaterMark

```
UBaseType_t uxTaskGetStackHighWaterMark( TaskHandle_t xTask );
```

返回栈使用空间最大的那次数值。

xTaskCallApplicationTaskHook

```
BaseType_t xTaskCallApplicationTaskHook( TaskHandle_t xTask,
                                         void *pvParameter );
```

执行相应的钩子函数。

xTaskGetIdleTaskHandle

```
TaskHandle_t xTaskGetIdleTaskHandle( void );
```

返回空闲任务的句柄。

xTaskGetIdleRunTimeCounter

```
TickType_t xTaskGetIdleRunTimeCounter( void );
```

返回空闲任务的运行时间。

xTaskNotify

```
BaseType_t xTaskNotify( TaskHandle_t xTaskToNotify,
                        uint32_t ulValue,
                        eNotifyAction eAction );
```

发送广播。

xTaskNotifyWait

```
BaseType_t xTaskNotifyWait( uint32_t ulBitsToClearOnEntry,
                            uint32_t ulBitsToClearOnExit,
                            uint32_t *pulNotificationValue,
                            TickType_t xTicksToWait );
```

等待广播。

xTaskNotifyStateClear

```
BaseType_t xTaskNotifyStateClear( TaskHandle_t xTask );
```

清除信号量。

SCHEDULER INTERNALS

- xTaskIncrementTick
- vTaskPlaceOnEventList
- vTaskPlaceOnUnorderedEventList
- vTaskPlaceOnEventListRestricted
- xTaskRemoveFromEventList
- vTaskRemoveFromUnorderedEventList
- vTaskSwitchContext
- uxTaskResetEventItemValue
- xTaskGetCurrentTaskHandle
- vTaskSetTimeOutState
- xTaskCheckForTimeOut
- vTaskMissedYield
- xTaskGetSchedulerState
- xTaskPriorityInherit
- xTaskPriorityDisinherit
- vTaskPriorityDisinheritAfterTimeout
- uxTaskGetTaskNumber
- vTaskSetTaskNumber
- vTaskStepTick
- eTaskConfirmSleepModeStatus
- pvTaskIncrementMutexHeldCount
- vTaskInternalSetTimeOutState