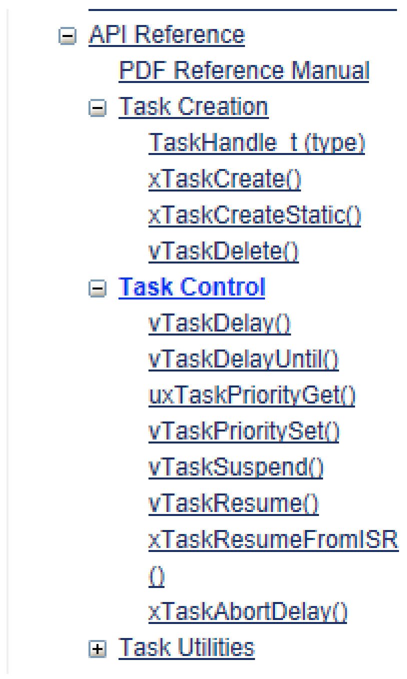
Lab Objective

To demonstrate FreeRTOS task management using the following features: -

- Task creation
- Task suspend and resume
- Task blocking using delays
- Task priority
- Task parameters

Reference: FreeRTOS Reference Manual Chapter 3

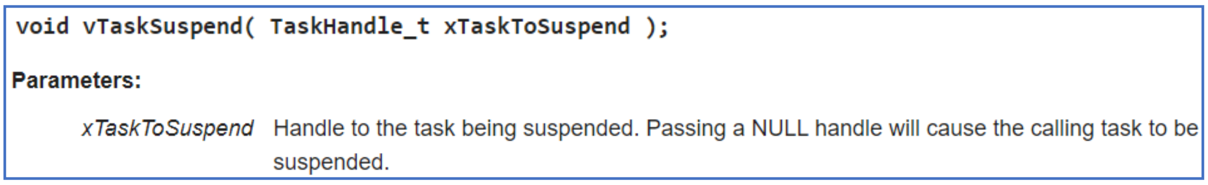


⊟ API Reference
  PDF Reference Manual
  ⊟ Task Creation
    TaskHandle_t (type)
    xTaskCreate()
    xTaskCreateStatic()
    vTaskDelete()
  ⊟ Task Control
    vTaskDelay()
    vTaskDelayUntil()
    uxTaskPriorityGet()
    vTaskPrioritySet()
    vTaskSuspend()
    vTaskResume()
    xTaskResumeFromISR()
    xTaskAbortDelay()
  ⊞ Task Utilities

Part A: Task Create and Delete

Create 2 tasks. Task 1 has higher priority than task 2. Task 1 is created in userApp(). Task 1 should run and then create task 2 and suspend itself. Task 2 then runs before deleting itself.

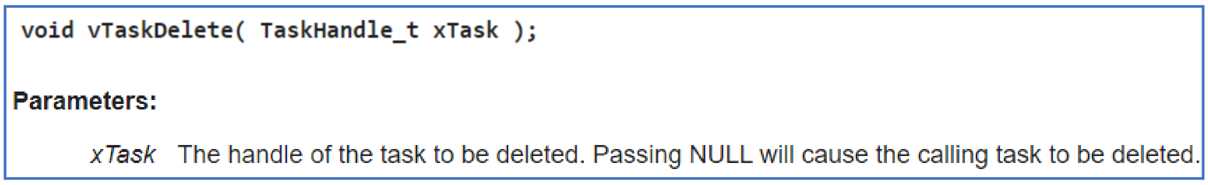
Use the xTaskCreate(), vTaskDelete() and the vTaskSuspend() API functions.



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

Parameters:

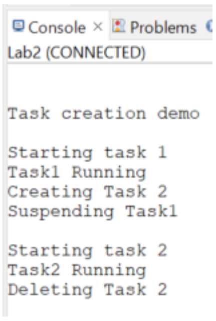
xTaskToSuspend	Handle to the task being suspended. Passing a NULL handle will cause the calling task to be suspended.



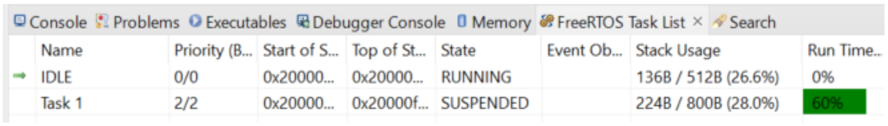
```
void vTaskDelete( TaskHandle_t xTask );
```

Parameters:

xTask	The handle of the task to be deleted. Passing NULL will cause the calling task to be deleted.



```
Lab2 (CONNECTED)


Task creation demo

Starting task 1
Task1 Running
Creating Task 2
Suspending Task1

Starting task 2
Task2 Running
Deleting Task 2
```



Name	Priority (B...	Start of S...	Top of St...	State	Event Ob...	Stack Usage	Run Time...
IDLE	0/0	0x20000...	0x20000...	RUNNING		136B / 512B (26.6%)	0%
Task 1	2/2	0x20000...	0x20000f...	SUSPENDED		224B / 800B (28.0%)	60%

<u>Part B: Task Priority</u>

Create 2 tasks in userApp(). Task 1 has priority 3 and task 2 has priority 1. Task 1 reads the priority of task 2 and increments it by 1. Task 2 should run after 2 iterations of task 1. Task 2 should then restore its priority to 1.

Edit the file FreeRTOSConfig.h to uncomment the line #define configUSE_TIME_SLICING 0. This ensures we are using pre-emptive only scheduling.

Use the uxTaskPriorityGet() and vTaskPrioritySet() API functions.

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

Parameters:

xTask	Handle of the task to be queried. Passing a NULL handle results in the priority of the calling task being returned.

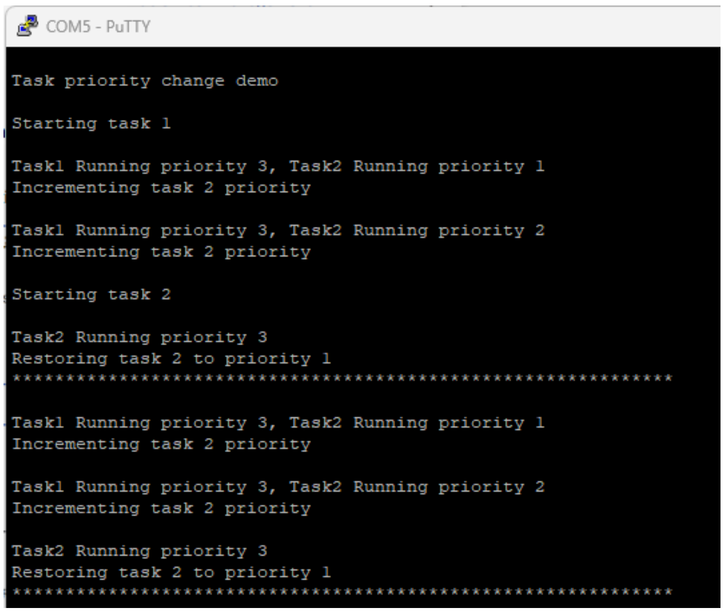
Returns:

The priority of xTask.

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

Parameters:

xTask	Handle of the task whose priority is being set. A NULL handle sets the priority of the calling task.
uxNewPriority	The priority to which the task will be set. Priorities are asserted to be less than `configMAX_PRIORITIES`. If `configASSERT` is undefined, priorities are silently capped at (`configMAX_PRIORITIES` - 1).

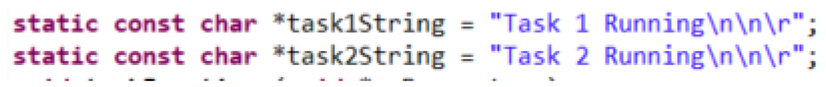


```
Task priority change demo

Starting task 1

Task1 Running priority 3, Task2 Running priority 1
Incrementing task 2 priority

Task1 Running priority 3, Task2 Running priority 2
Incrementing task 2 priority

Starting task 2

Task2 Running priority 3
Restoring task 2 to priority 1
**************************************************************

Task1 Running priority 3, Task2 Running priority 1
Incrementing task 2 priority

Task1 Running priority 3, Task2 Running priority 2
Incrementing task 2 priority

Task2 Running priority 3
Restoring task 2 to priority 1
**************************************************************
```

Part C: Task Parameter Passing

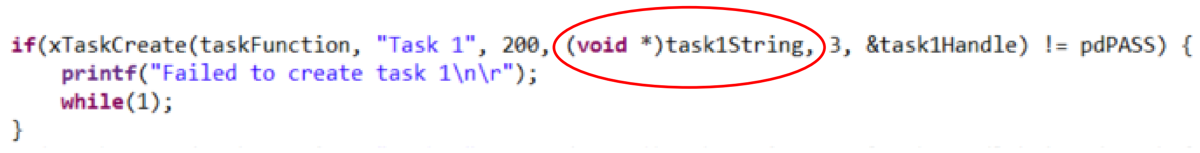
Create 2 tasks of the same priority. Both tasks should use the same task function. The task function should simply print a string associated with the current running task. The task function should then delay for 1 second. Use the vTaskDelay() API function.

Create a global string pointer for each task.



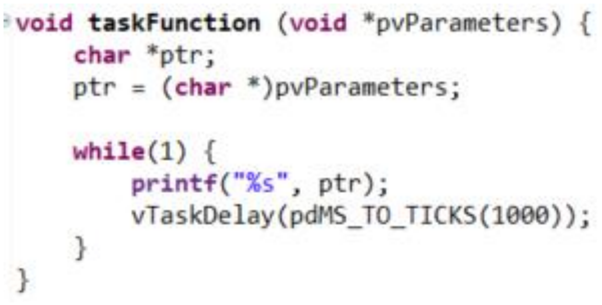
```
static const char *task1String = "Task 1 Running\n\n\r";
static const char *task2String = "Task 2 Running\n\n\r";
```

The task create function is passed a void pointer to the task string.



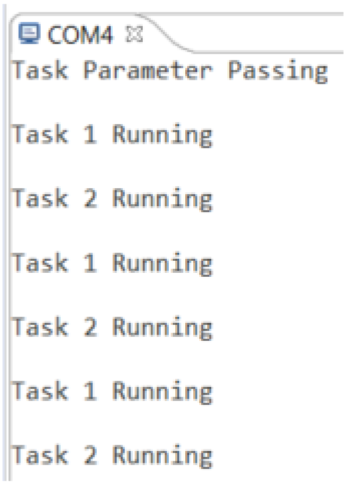
```
if(xTaskCreate(taskFunction, "Task 1", 200, (void *)task1String, 3, &task1Handle) != pdPASS) {
    printf("Failed to create task 1\n\r");
    while(1);
}
```

The task function then casts the passed parameter to a char pointer.



```
void taskFunction (void *pvParameters) {
    char *ptr;
    ptr = (char *)pvParameters;

    while(1) {
        printf("%s", ptr);
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
```

Reference: Section 3.4 in Mastering the FreeRTOS Real Time Kernel. Tutorial



```
Task Parameter Passing

Task 1 Running

Task 2 Running

Task 1 Running

Task 2 Running

Task 1 Running

Task 2 Running
```