Real Time Operating System
"FreeRTOS"
Change Task Priority

Sherif Hammad

Using the FreeRTOS Real Time Kernel - a Practical Guide - Cortex M3 Edition (FreeRTOS Tutorial Books)

by Richard Barry

Changing task priorities; Example 8

```
/* Declare a variable that is used to hold the handle of Task 2. */
xTaskHandle xTask2Handle;

int main( void )
{
    /* Create the first task at priority 2.  The task parameter is not used
    and set to NULL.  The task handle is also not used so is also set to NULL. */
    xTaskCreate( vTask1, "Task 1", 240, NULL, 2, NULL );
    /* The task is created at priority 2 _____^. */

    /* Create the second task at priority 1 - which is lower than the priority
    given to Task 1.  Again the task parameter is not used so is set to NULL -
    BUT this time the task handle is required so the address of xTask2Handle
    is passed in the last parameter. */
    xTaskCreate( vTask2, "Task 2", 240, NULL, 1, &xTask2Handle );
    /* The task handle is the last parameter ____^^^^^^^^^^^^^ */

    /* Start the scheduler so the tasks start executing. */
    vTaskStartScheduler();

    /* If all is well then main() will never reach here as the scheduler will
    now be running the tasks.  If main() does reach here then it is likely that
    there was insufficient heap memory available for the idle task to be created.
    Chapter 5 provides more information on memory management. */
    for( ;; );
}
```

Listing 24. The implementation of main() for Example 8

Changing task priorities; Example 8

```
void vTask1( void *pvParameters )
{
unsigned portBASE_TYPE uxPriority;

    /* This task will always run before Task 2 as it is created with the higher
    priority.  Neither Task 1 nor Task 2 ever block so both will always be in either
    the Running or the Ready state.

    Query the priority at which this task is running - passing in NULL means
    "return my priority". */
    uxPriority = uxTaskPriorityGet( NULL );

    for( ;; )
    {
        /* Print out the name of this task. */
        vPrintString( "Task 1 is running\n" );

        /* Setting the Task 2 priority above the Task 1 priority will cause
        Task 2 to immediately start running (as then Task 2 will have the higher
        priority of the two created tasks).  Note the use of the handle to task
        2 (xTask2Handle) in the call to vTaskPrioritySet().  Listing 24 shows how
        the handle was obtained. */
        vPrintString( "About to raise the Task 2 priority\n" );
        vTaskPrioritySet( xTask2Handle, ( uxPriority + 1 ) );

        /* Task 1 will only run when it has a priority higher than Task 2.
        Therefore, for this task to reach this point Task 2 must already have
        executed and set its priority back down to below the priority of this
        task. */
    }
}
```

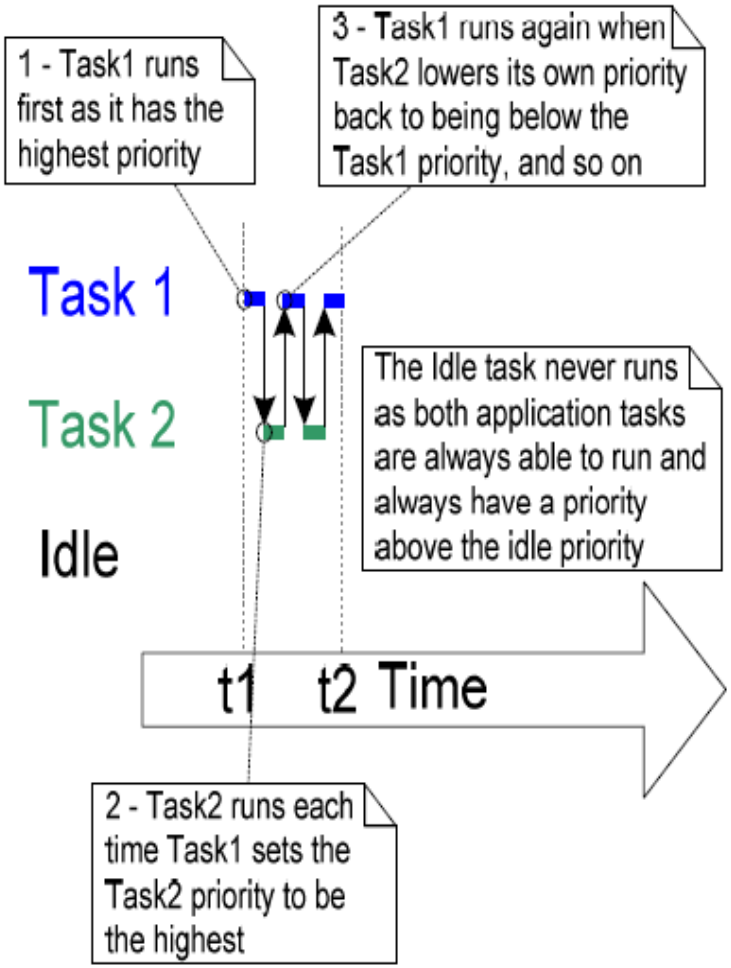
Listing 22. The implementation of Task 1 in Example 8

Changing task priorities; Example 8

```
void vTask2( void *pvParameters )
{
unsigned portBASE_TYPE uxPriority;

    /* Task 1 will always run before this task as Task 1 is created with the
    higher priority.  Neither Task 1 nor Task 2 ever block so will always be
    in either the Running or the Ready state.

    Query the priority at which this task is running - passing in NULL means
    "return my priority". */
    uxPriority = uxTaskPriorityGet( NULL );

    for( ;; )
    {
        /* For this task to reach this point Task 1 must have already run and
        set the priority of this task higher than its own.

        Print out the name of this task. */
        vPrintString( "Task2 is running\n" );

        /* Set our priority back down to its original value.  Passing in NULL
        as the task handle means "change my priority".  Setting the
        priority below that of Task 1 will cause Task 1 to immediately start
        running again - pre-empting this task. */
        vPrintString( "About to lower the Task 2 priority\n" );
        vTaskPrioritySet( NULL, ( uxPriority - 2 ) );
    }
}
```

Listing 23. The implementation of Task 2 in Example 8

Changing task priorities; Example 8



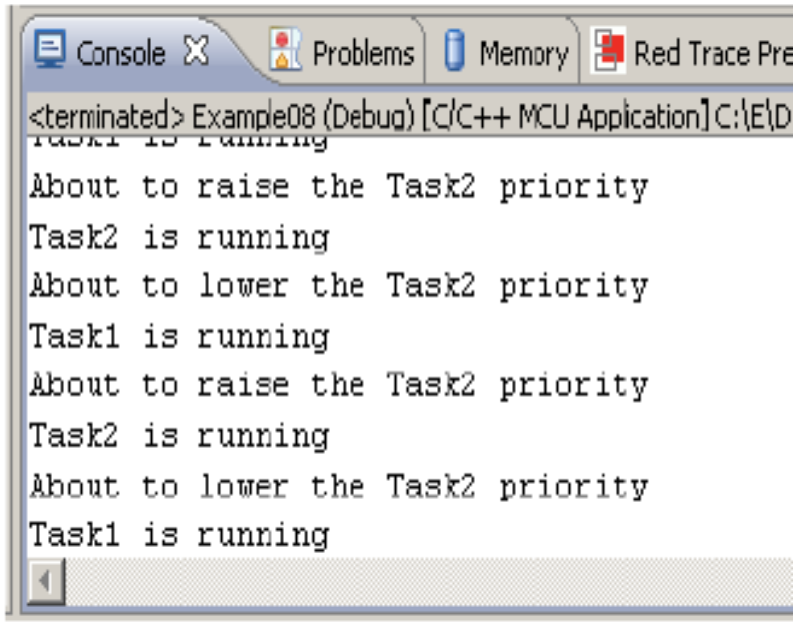


Figure 14. The sequence of task execution when running Example 8

The Scheduling Algorithm—A Summary

- **Fixed Prioritized Pre-emptive Scheduling**
  - **Each task is assigned a priority.**
  - **Each task can exist in one of several states.**
  - **Only one task can exist in the Running state at any one time.**
  - **The scheduler always selects the highest priority Ready state task to enter the Running state.**

- **Fixed Priority' because each task is assigned a priority that is not altered by the kernel itself (only tasks can change priorities);**

- **'Pre-emptive' because a task entering the Ready state or having its priority altered will always pre-empt the Running state task, if the Running state task has a lower priority.**

The Scheduling Algorithm—A Summary

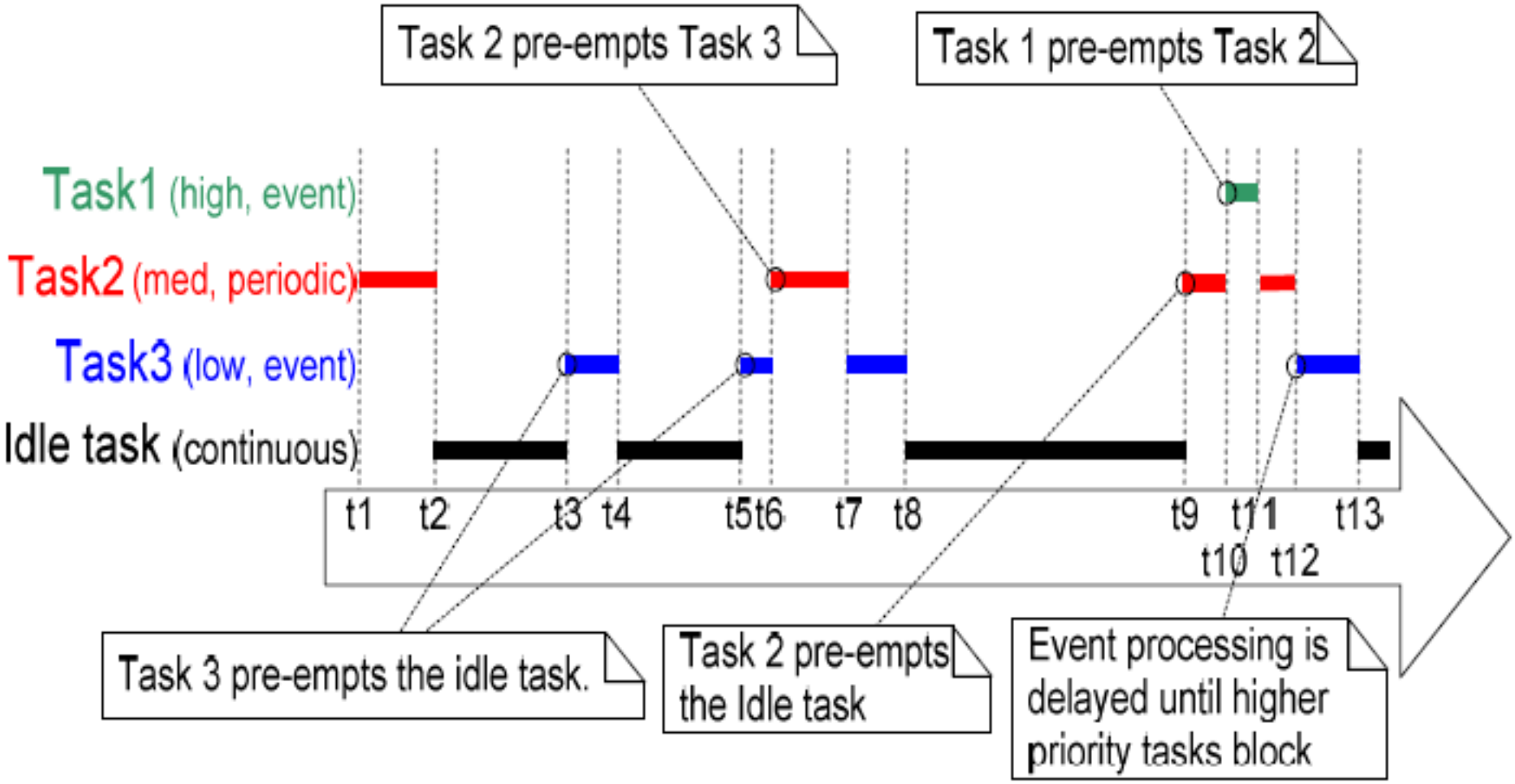


Figure 18. Execution pattern with pre-emption points highlighted