UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Máster Universitario en Ingeniería de Telecomunicación

Tarea 4 - III
Estadísticas en uCOS-II. Tamaños de pila y tiempos en RTOS

CHS

Autor:
Andrés Ruz Nieto
Gerardo Arias Martínez

VALENCIA, 2022

TELECOM UPV VLC

Al crear la función *OSTaskStatHook* personalizada, se tendrá que comentar la función original que se encuentra en *os_cpu_c.c*

En ucosii_stat_MTL.c se puede ver que hay 6 tareas programadas, 4 tareas especificas y 2 "especiales", de estas 2, una se encargará de crear el resto de tareas y la segunda se encarga de calcular las estadísticas.

La función OSStatInit que se encuentra en la tarea de inicialización sirve para habilitar a uC para habilitar las estadísticas.

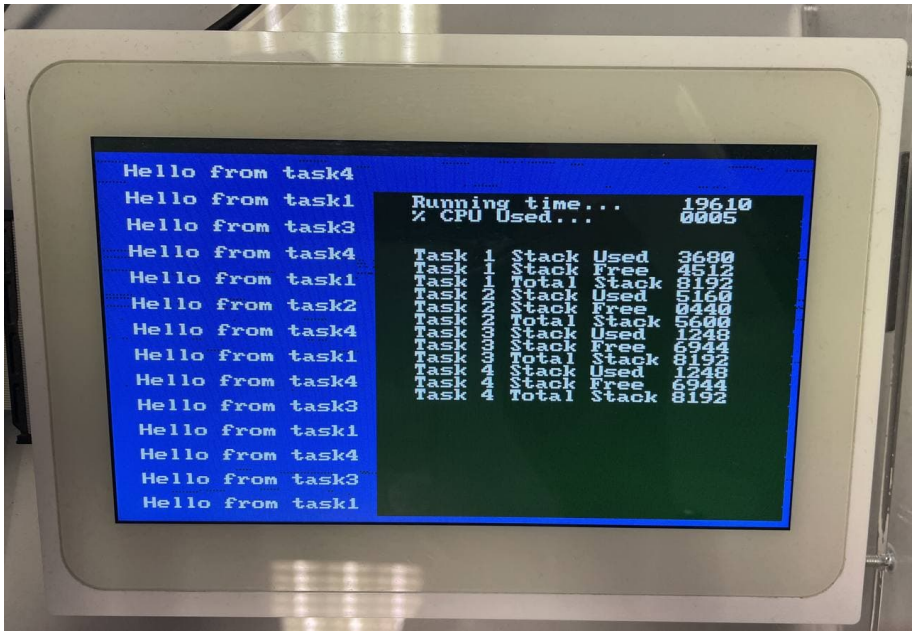
Cuando se crean las tareas hay que añadir el parámetro *OS_TASK_OPT_STK_CHK* y *OS_TASK_OPT_STK_CLR* para consultar y poder borrar la pila.

La tarea *showstats* bloquea el scheduler con OSSchedLock nada más empezar para evitar que otras tareas expulsen a esta. Cada segundo mostrará las estadísticas. Realizará un check de la pila de cada tarea.

Ejercicio 1

```
El mensaje es: Hola Don PepitoHello from task1

Hello from task1, I need 52198 - 51389 = 809 ticks from RTOS
1 mensaje desde task2 es: Hola Don José
```



Las tareas nada más empezar almacena el tiempo en el que se encuentra a través de *OSTimeGet* y al finalizar vuelve capturar el tiempo en otra variable, a continuación

se realizará una resta de ambas variables para obtener los ticks que ha tardado en ejecutarse esa tarea.

Como se puede ver la *tarea1* está usando el $44{,}9\,\%$ de la pila, la *tarea2* el $92\,\%$, *la tarea3* el $15\,\%$ y, por último, *la tarea4* el $16\,\%$

```
El mensaje es: Hola Don Pepito
El mensaje es: Hola Don Pepito
El mensaje es: Hola Don Pepito
El
TaskExecTime      Task 1   is   0
TaskTotalExecTime Task 1   is   0
TaskCtr   of      Task 1   is   0

TaskExecTime      Task 2   is   0
TaskTotalExecTime Task 2   is   0
TaskCtr   of      Task 2   is   0

TaskExecTime      Task 3   is   0
TaskTotalExecTime Task 3   is   0
TaskCtr   of      Task 3   is   0

TaskExecTime      Task 4   is   0
TaskTotalExecTime Task 4   is   0
TaskCtr   of      Task 4   is   0
 mensaje es: Hola Don Pepito
El mensaje es: Hola Don Pepito
El mensaje es: Hola Don Pepito
El mensaje es: Hola Don Pepito
El mensaje es: Hola Don Pepito
```

Aquí se muestra un 0 porque aún queda por realizar una consulta para mostrar esta estadística correctamente. TaskCtr es un contador del número de veces que una tarea ha entrado y salido. TaskExecTime es el tiempo de ejecución de la tarea y TaskTotExecTime es el tiempo acumulado.

Para ello se habilitará *OSTaskSwHook*, al habilitar esta función el *Running Time* dejará de funcionar, ya que *OSTaskSwHook* (se ejecuta cada vez que hay un cambio de tareas) pone a 0 el *Running Time* (*OsTimeSet(0)*)

Cada tick equivale a 1/1000s declarado en system.h

```
TaskExecTime      Task 1   is   0
TaskTotalExecTime Task 1   is 117
TaskCtr   of      Task 1   is 125

TaskExecTime      Task 2   is   1
TaskTotalExecTime Task 2   is 146
TaskCtr   of      Task 2   is 310

TaskExecTime      Task 3   is   0
TaskTotalExecTime Task 3   is   0
TaskCtr   of      Task 3   is   3

TaskExecTime      Task 4   is   0
TaskTotalExecTime Task 4   is   0
TaskCtr   of      Task 4   is   4
```

El task_stacksize se define el palabras largas (4·bytes).

Vamos a realizar un desbordamiento de la pila de la tarea 2, cambiando el tamaño de su pila. Al hacer esto se puede ver que el programa deja de funcionar, dependiendo de lo grande que sea el desbordamiento el procesador realizará unas cosas u otras (se reinicia, se cuelga...)