Labo de Micros - Guía 11: RTOS

1. Estudiar en profundidad el código de ucosiii\_labo\_de\_micros\_project.

a. ¿Cuántos threads hay corriendo, luego de la inicialización?

**OS\_IdleTask(), OS\_TickTask(), OS\_StatTask() and OS\_TaskTmr()**

**"App Start Task" que fue creada en el main antes de OSStart y "Task 2" que fue creada durante el inicio de la ejecución de la Start Task.**

b. ¿Qué tarea hacen esos threads?

**Start Task crea la Task 2 y luego hace titilar el led verde cada 1 segundo.**

**Task 2 hace titilar el led rojo cada 500 milisegundos, y cada vez que togglea también emite por un semáforo la señal OS\_OPT\_POST\_1 (Post message to highest priority task waiting).**

c. ¿Qué hacen los threads la mayor parte del tiempo?

**La mayor parte del tiempo no hacen nada, esperan a que terminen los timers.**

d. ¿Qué thread usa la CPU la mayor parte del tiempo (~99%)?

**Supongo que OS\_IdleTask(). Debería chequearese con OSTaskStkChk().**

2. Estudiar en profundidad la estructura del proyecto ucosiii\_labo\_de\_micros\_project.

a. Observar el código en source\rtos\uCOSIII\src\uCOS-III\Ports\ARM-

Cortex-M4\Generic\GNU\os\_cpu\_c.c. ¿Qué timer de hardware se usa para el tick() del RTOS? ¿Puede usarse ese timer para los drivers?

**El RTOS usa el SysTick (System Tick, línea 420 del archivo os\_cpu\_c.c)**

**Creo que el timer puede usarse para los drivers redefiniendo la función OSTimeTickHook que es llamada desde la ISR del SysTick (a chequear).**

b. Observar el header source\ucosiii\_config\os\_cfg\_app.h. ¿Qué frecuencia tiene el tick() del RTOS?

**La frecuencia es 1000 Hz, el periodo es 1 milisegundo**

**#define OS\_CFG\_TICK\_RATE\_HZ 1000u // Tick rate in Hertz (10 to 1000 Hz)**

c. Intentar entender las configuraciones en los headers que están en source\ucosiii\_config, usando el manual del RTOS: (<https://doc.micrium.com/display/kernel304/uC-OS-III+Documentation+Home>)

3. Abrir la documentación de uC/OS III y leer el capítulo “Getting Started with uC/OS III”.

**Notas:**

**Texto

Descripción generada automáticamente**

**El mensaje que se transmite/recibe a través de una cola de mensajes es un “puntero a algo”.** **Both the sender and receiver must agree as to the meaning of the message.**

4. Crear un nuevo Thread (Thread3) que utilice el LED azul, titilando cada 100ms, usando OSTimeDly en lugar de OSTimeDlyHMSM. ¿Qué diferencia hay entre las dos?

**La diferencia es que OSTimeDly se especifica en Ticks, mientras que la otra con horas, minutos, segundos, milisegundos. Como la duración de los ticks es 1ms entonces es casi igual.**

5. En lugar de usar un delay para hacer titilar el LED azul, usar el semáforo (semExample) para hacer que el Thread2 le mande una señal (OSSemPost) al Thread3, cada 500ms. El Thread3 quedará suspendido (OSSemPend) hasta recibir la señal, luego hará un toggle del LED, y volverá a dormir.

**static** **void** **Task3**(**void** \*p\_arg) {

(**void**)p\_arg;

OS\_ERR os\_err;

**while** (1) {

OSSemPend(&semTest, 0, OS\_OPT\_PEND\_BLOCKING, NULL, &os\_err);

LED\_B\_TOGGLE();

}

}

6. En lugar de usar un semáforo para hacer titilar el LED azul, implementar una MessageQueue para hacer que el Thread2 le mande un mensaje (OSQPost) al Thread3, cada 500ms. El Thread 2 le indicará el color del LED a togglear, mediante el mensaje. El Thread3 quedará suspendido (OSQPend) hasta recibir el mensaje, luego hará un toggle del LED que corresponda, y volverá a dormir.

7. Conectar la placa del TP1. Identificar las modificaciones necesarias para integrar  
el código del TP1 en el proyecto:

a. Si el TP1 usaba SysTick, ¿a qué frecuencia?

**A la misma frecuencia: 1ms**

b. ¿Se puede “hookear” el tick() que usa el RTOS?

**Sí, mediante la definición de OS\_AppTimeTickHookPtr (manual página 559).**

**Archivo: os\_cpu\_c.c**

c. Abrir SDK\startup\startup\_mk64f12.c.

i. Identificar la tabla de vectores de interrupción. ¿Qué código se  
ejecuta inmediatamente luego del reset?  
ii. Seguir la secuencia (sin saltear pasos) desde el reset hasta el main.  
iii. ¿El main arranca con las interrupciones habilitadas o  
deshabilitadas?

d. Copiar el código del main() del TP1 al ThreadStart del ejemplo. Quitar la  
inicialización del Thread2 (OSTaskCreate(&Task2TCB, …)). ¿El TP1  
funciona?

**Yeah**

e. Si no funciona, hacer que funcione antes de seguir.

8. Delinear un plan para:  
a. Modificar el driver de la tarjeta magnética para que señalice, mediante un  
semáforo (OSSemPost) cuando hay un ID de tarjeta leído.  
b. Modificar el driver del encoder para que señalice, mediante un semáforo  
(OSSemPost) cuando hay un evento de encoder.  
c. Modificar el main para evitar pollear al driver de la tarjeta y del encoder, y  
en su lugar usar OSPendMulti().  
d. Re-activar el Thread2 e implementar una MessageQueue.  
e. Postear en la MessageQueue un mensaje, cada vez que hay un ID leído.  
f. En el Thread2, utilizar OSQPend() para esperar un mensaje, y togglear un  
LED cada vez que se reciba uno.