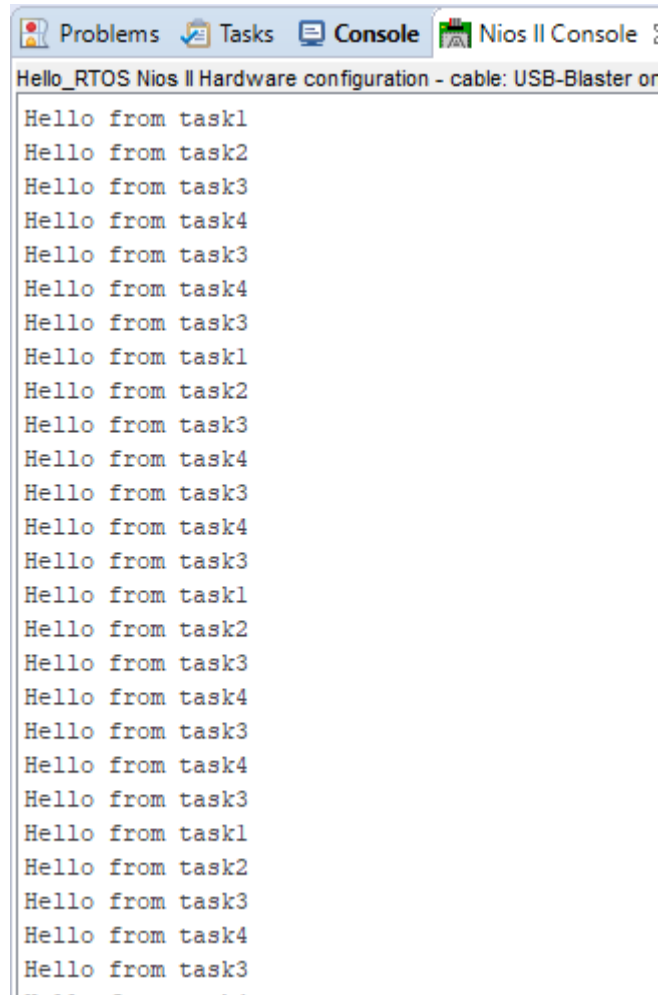


Practica 1 RTOS

Jose Luis Rocabado Rocha

1. Ejercicio 1

Sale esto. Task 3 cada 1 segundos, task4 cada 1.5 segundos.



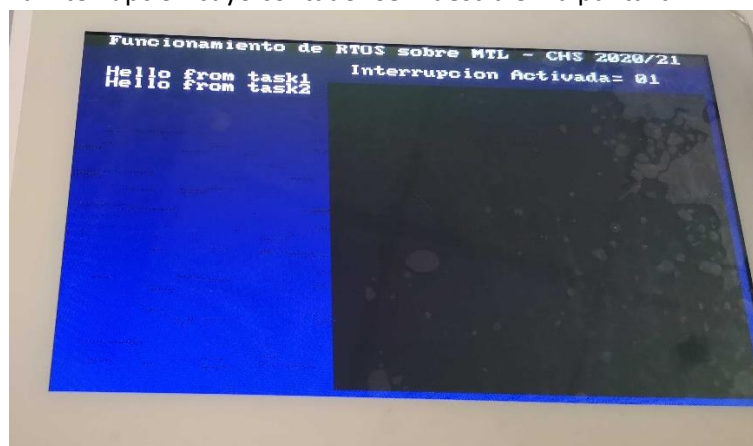
The screenshot shows a Nios II Console window with the title bar 'Hello_RTOS Nios II Hardware configuration - cable: USB-Blaster or'. The console output displays a sequence of messages from four tasks: task1, task2, task3, and task4. The messages are interleaved, showing task3 and task4 executing more frequently than task1 and task2. The output is as follows:

```
Hello from task1
Hello from task2
Hello from task3
Hello from task4
Hello from task3
Hello from task4
Hello from task3
Hello from task1
Hello from task2
Hello from task3
Hello from task4
Hello from task3
Hello from task4
Hello from task3
Hello from task1
Hello from task2
Hello from task3
Hello from task4
Hello from task3
Hello from task4
Hello from task3
Hello from task1
Hello from task2
Hello from task3
Hello from task4
Hello from task3
```

2. Ejercicio 2

¿Qué ocurre cuando se utiliza el pulsador KEY[1]?

Se genera una interrupción cuyo contador se muestra en la pantalla MTL.



¿Qué papel desempeñan las funciones de manejo de LEDs?

Nos permiten comprobar de forma visual la temporización de las tareas.

¿Y las de MTL?

Visualizar el curso de la ejecución sin consumir demasiados recursos de la cpu.

¿Qué problemas pueden existir con el debug por printf?

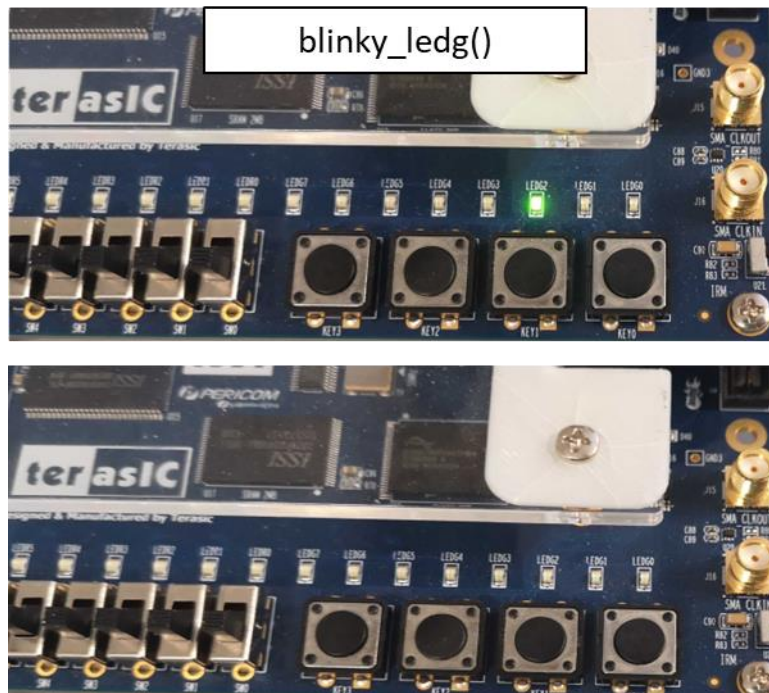
Pues que el printf consume muchos recursos para realizar la comunicación con el JTAG.

¿Existe una buena comunicación entre tareas? ¿Qué método de comunicación se usa?

No es la mejor comunicación puesto que el sistema es expulsivo y podría producir conflictos al acceder a la variable global "líneas".

3. Ejercicio 3

La función "*blinky_ledg()*" nos permite realizar un *toggle* del LEDG2 cada vez que se ejecuta la tarea 3. El parpadeo producido en las otras tareas nos permite observar el tiempo durante el que se ejecutan sus tareas dado que al inicio de la tarea se encienden y al final se apagan sus respectivos LEDs.



4. Ejercicio 4

Snippet de código, se observa que se puede cambiar tanto la prioridad de las otras tareas como la propia de la tarea.

```
321 void task3(void* pdata)
322 {
323     char visualiza_string[40] = "Hello from task3";
324     static int posicion=0;
325     while (1)
326     {
327         blinky_ledg (green_LED_ptr, 2);
328 #ifdef PRINT
329         printf("%s\n",visualiza_string);
330 #endif
331         posicion=(linea>ultima_linea)?primera_linea:linea++;
332         if (linea>ultima_linea){linea=primera_linea;}
333         MTL_text (2, posicion, visualiza_string);
334         borra_lineas_pantalla(posicion+1,posicion+2);
335
336         OSTaskChangePrio (TASK3_PRIORITY, BAD_PRIORITY);
337         //OSTimeDlyHMSM(0, 0, 2, 0);
338
339         //OSTaskSuspend(OS_PRIO_SELF);
340         OSTaskSuspend(TASK2_PRIORITY);
341
342         if (OSTaskDelReq(OS_PRIO_SELF) == OS_ERR_TASK_DEL_REQ) {
343             /* Release any owned resources; */
344             /* De-allocate any dynamic memory; */
345             printf("Sayonara \n");
346             OSTaskDel(OS_PRIO_SELF);
347         }
348
349         OSTimeDlyHMSM(0, 0, 0, 500);
350     }
351 }
```

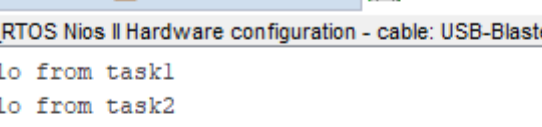
5. Ejercicio 5

Se comprueba que al tener una prioridad muy mala pero mismo tiempo de ejecución se ejecuta después de la tarea 1 y 2.

```
321 void task3(void* pdata)
322 {
323     char visualiza_string[40] = "Hello from task3";
324     static int posicion=0;
325     while (1)
326     {
327         blinky_ledg (green_LED_ptr, 2);
328 #ifdef PRINT
329         printf("%s\n",visualiza_string);
330 #endif
331         posicion=(linea>ultima_linea)?primera_linea:linea++;
332         if (linea>ultima_linea){linea=primera_linea;}
333         MTL_text (2, posicion, visualiza_string);
334         borra_lineas_pantalla(posicion+1,posicion+2);
335
336         OSTaskChangePrio (TASK3_PRIORITY, BAD_PRIORITY);
337         //OSTimeDlyHMSM(0, 0, 2, 0);
338
339         //OSTaskSuspend(OS_PRIO_SELF);
340         OSTaskSuspend(TASK2_PRIORITY);
341
342         if (OSTaskDelReq(OS_PRIO_SELF) == OS_ERR_TASK_DEL_REQ) {
343             /* Release any owned resources; */
344             /* De-allocate any dynamic memory; */
345             printf("Sayonara \n");
346             OSTaskDel(OS_PRIO_SELF);
347         }
348
349         OSTimeDlyHMSM(0, 0, 0, 500);
350     }
351 }
```

6. Ejercicio 6

Tras su primera ejecución, la tarea 3 se suspende dado que se ha usado la macro "OS_PRIO_SELF".



The screenshot shows the Nios II IDE interface. The top toolbar includes icons for Problems, Tasks, Console, and Nios II Console. The Console window is active, displaying the output of a program. The output text is as follows:

```
Hello_RTOS Nios II Hardware configuration - cable: USB-Blaster on loc  
Hello from task1  
Hello from task2  
Hello from task3  
Hello from task1  
Hello from task1  
Hello from task1  
Hello from task1  
Hello from task2  
Hello from task1  
Hello from task1  
Hello from task1  
Hello from task2
```

Sucede lo mismo, pero con la tarea 2 una vez se ejecuta la tarea 3.
Se demuestra así que se puede suspender tareas de mayor prioridad dentro de tareas de menor prioridad.

[illegible]

Se suspende la tarea 2 tras ejecutar la tarea 3 y no se vuelve a activar hasta que se ejecute la tarea 4 que la reactiva. Sin embargo, tras ejecutarse la tarea 3 se vuelve a suspender.

[illegible]

8. Ejercicio 8

Lo mismo que en el ejercicio anterior, sin embargo, tras ejecutarse la tarea 4 se ponen en modo *RUNNING* las demás tareas.

Problems Tasks Console Nios II Console

Hello_RTOS Nios II Hardware configuration - cable: USB-Blaster on loc

```
Hello from task1  
Hello from task2  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from recovery  
Hello from task2  
Hello from task1  
Hello from task2  
Hello from task3  
Hello from task1  
Hello from task3
```

9. Ejercicio 9

Tras ejecutar la tarea 4 se “duerme” definitivamente la tarea 3 de forma directa.

Problems Tasks Console Nios II Console

Hello_RTOS Nios II Hardware configuration - cable: USB-Blaster on loca

```
Hello from task1  
Hello from task2  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from task3  
Hello from task1  
Hello from recovery  
Hello from task2  
Hello from task1  
Hello from task2  
Hello from task1  
Hello from task1  
Hello from task1  
Hello from task2
```

10. Ejercicio 10

Sin embargo, si se utiliza la solicitud de eliminar tarea, se ejecuta el código que imprime “sayonara” antes de “dormir definitivamente la tarea 3”

