SISTEMAS BASADOS EN MICROPROCESADOR
PRÁCTICA 3. MULTITAREA COOPERATIVA

Grado en Ingeniería Electrónica Industrial (3er curso)
Curso 2024/2025

1 INTRODUCCIÓN

Una forma rudimentaria para realizar una multiprogramación es la multitarea cooperativa, tal como se ha descrito en teoría. Esta práctica se va a centrar en aprovechar parte de las subrutinas de las anteriores sesiones para desarrollar un programa que ejecute tres procesos independientes con una temporización precisa: una animación de leds, otra de los segmentos de la batería del LCD y una visualización de dígitos en el display de 7 segmentos.

1.1 Objetivos

- Establecer un sistema de temporización precisa.
- Abordar un sistema de multitarea cooperativa sencillo con tres procesos.

2 ESTUDIO TEÓRICO

Busque y estudie en la documentación de examen de la página web de la asignatura la documentación del Timer A (`TimerA.pdf`). En ella se recogen los registros de control de los distintas instancias del módulo Timer A del MSP430FR6989. Estudie también los conceptos de multitarea cooperativa.

3 ESTUDIO PRÁCTICO

3.1 Preparando el proyecto

Copie el proyecto de la práctica anterior y llámelo P3.1. Se van a reaprovechar todos sus módulos, así como parte del programa principal en C.

3.2 Módulo de gestión del SytemTimer `st.asm`

Se necesitará añadir un nuevo fichero en ensamblador al proyecto. Puede partir de un fichero vacío o copiar uno de los disponibles (`lcd.asm`, `cs.asm`). Llámelo `st.asm`. Si ha copiado uno anterior, borre el código y las etiquetas de la directiva `.global`.No olvide crear también el fichero de cabecera asociado `st.h` en el que incluirá los prototipos de las funciones de este módulo y que deberá incluir en cualquier fichero que usesus servicios.

Defina una variable que albergará el valor actual del tiempo del sistema en TICS de reloj y no la incluya en la lista de símbolos publicados (`.global`):

```
.bss    SystemTimer, 4
```

También se necesita una variable para almacenar el valor inicial del CCR0, que también es el periodo, ya que el timer se va a configurar en modo continuo:

```
.bss    stPeriodo, 2
```

Los prototipos de este módulo son:

3.2.1 Inicialización `stIni`

Inicializa el *system timer*. Para ello configura el TimerA2 en modo `CONTINUOUS` a 32768Hz (fuente ACLK y divisor 1). `periodo` es el valor a cargar en CCR0 que va a determinar el valor de la interrupción periódica. Dado que estamos en modo continuo, la subrutina deberá guardar localmente el `periodo` en `stPeriodo` para posteriores actualizaciones:

```
void stIni (unsigned int periodo);
```

3.2.2 Lectura del tiempo `stTime`

Devuelve el valor actual del *system timer*. Dado que la variable ocupa dos palabras, la lectura no es atómica[1]. Para evitar condiciones de carrera[2] con la interrupción periódica, asegúrese que las mismas no ocurren entre las dos lecturas:

```
unsigned long stTime (void);
```

3.2.3 Comparación de tiempo `stComp`

Compara tiempo con el *system timer*. Devuelve un número negativo si `tiempo<ST`, positivo si `tiempo>ST` y 0 si `tiempo=ST`:

```
int stComp (unsigned long tiempo);
```

3.2.4 Subrutina de servicio de interrupción `stA2ISR`

La subrutina de servicio de interrupción también se definirá en este módulo (fichero), pero no se publicará para que no sea accesible desde el exterior (es decir, no irá en `.global`). El cometido de la interrupción será incrementar la variable `SystemTimer`. También deberá actualizar el valor del CCR0 para que la próxima IRQ se produzca en tiempo y forma.

3.3 Programa principal en C

A modo de demostración, haga un programa que configure los módulos adecuadamente y gestione un *superbucle* con tres tareas cooperativas:

- Animación de la pantalla LCD. Misma funcionalidad que en la práctica 2, pero sustituyendo la temporización mediante un bucle con espera activa, por el nuevo método de tareas periódicas y mostrando su nombre y dos apellidos. Frecuencia de la tarea, 2 Hz.
- Animación de los segmentos de capacidad de batería del LCD (B1 a B6) de forma que inicialmente se encienda B1, pasado un tiempo, se apague B1 y se encienda B2 y así sucesivamente hasta B6. Una vez terminado el barrido de izquierda a derecha, se repetirá de derecha a izquierda. Después repita la animación al completo. Frecuencia de la tarea: 8 Hz.
- Animación de los leds de usuario del Launchpad. En cada fase de animación, un led estará encendido y el otro apagado. Al cambiar de fase, se invertirá el estado de cada led. Frecuencia de la tarea: 1 Hz.

---

1. Una operación se llama atómica cuando no puede ser interrumpida. Una instrucción siempre es atómica.
2. Una condición de carrera (*race condition*) ocurre cuando dos o más procesos acceden un recurso compartido sin control, de manera que el resultado combinado de este acceso depende del orden de llegada.

3.3.1 Reduciendo el consumo

Una vez que se ha dado una pasada al superbucle, no hay nada que hacer hasta que se incremente el `SystemTimer`. Por ello, lo mejor es dormir la CPU y entrar en un modo de bajo consumo. Dado que se requiere el ACLK activo para alimentar el TA2, no se puede ir más allá de LPM3. Modifique el superbucle para entrar en LPM3 al final del mismo. También tendrá que modificar la ISR del `TimerA2` para que el micro se despierte tras la ISR y pueda volver al superbucle para dar atención a las tareas cooperativas.