Práctica 3

Estructura de Computadores

Modos de ejecución, gestión de excepciones y entrada/salida

Práctica 3

E/S por interrupciones no vectorizadas

Ejecutando el código de la práctica 2 hemos podido observar que a veces no se atiende adecuadamente a los dispositivos debido a la espera software introducida.

En la práctica 3 modificaremos el código anterior para que funcione por interrupciones no vectorizadas por la línea IRQ del procesador. La idea es la siguiente:

- Haremos que el movimiento del led alrededor del display de 8 segmentos esté gobernado por una interrupción periódica (de periodo 2s) generada por el **TIMER0.**

- Haremos que los botones sean atendidos bajo interrupción.

- La función loop no tendrá, por tanto, nada que hacer.

Para que funcione el proyecto **añadimos al proyecto dos nuevos ficheros**:

- **intcontroller.h** e **intcontroller.c**: interfaz e implementación de funciones del módulo que maneja el controlador de interrupciones. El alumno deberá completar la implementación de todas las funciones de este módulo, siguiendo las indicaciones de los comentarios y la documentación de [TPGb, um-].

- **timer.h** y **timer.c**: interfaz e implementación de funciones del módulo que maneja los temporizadores PWM. El alumno deberá completar la implementación de todas las funciones de este módulo, siguiendo las indicaciones de los comentarios y la documentación de [TPGb, um-].

En el fichero main.c deberemos realizar los siguientes cambios:

- Función **setup**:
    - Pulsadores: debemos cambiar la configuración del puerto G para que los pines 6 y 7 activen las señales EINT6 y EINT7 respectivamente, utilizando el interfaz del puerto G definido en **gpio.h**.

    - TIMER0: debemos configurarlo para que genere interrupciones periódicas, con un periodo de 2 segundos, utilizando el interfaz definido en **timer.h**.

    - Debemos configurar el controlador de interrupciones para que active la línea IRQ en modo no vectorizado y deje la línea FIQ deshabilitada, configure las líneas **TIMER0** y **EINT4567** por la línea IRQ del procesador y las deje habilitadas.

- Función **timer_ISR**: a pesar del nombre, esta función no es propiamente una rutina de tratamiento de interrupción, ya que es invocada desde **irq_ISR**. Debe declararse como una rutina corriente. Es la encargada de mover el led en el display de 8 segmentos una posición. La dirección del movimiento será la que esté almacenada en la variable **RL**.

- Función **button_ISR**: a pesar del nombre, esta función no es propiamente una rutina de tratamiento de interrupción, ya que es invocada desde **irq_ISR**. Debe declararse como una rutina corriente. Es la encargada de saber qué botón se ha pulsado consultando el registro **EXTINTPND** y realizar las tareas asociadas a la pulsación de cada uno de los botones (encender o apagar un led, cambiar la dirección de giro del *led rotante* y parar o arrancar el timer).

- Se elimina la función **loop**

En el fichero **init.asm** deberemos añadir:

- La dirección de comienzo de la rutina **irq_ISR** en la tabla de direcciones de las rutinas de tratamiento de excepciones.

- Función **irq_ISR (completarla en ensamblador).** Esta rutina será declarada como rutina de tratamiento de interrupciones IRQ. Esta rutina hará una encuesta accediendo al registro **ISPR** del controlador de interrupciones:
  - Si es una interrupción por la línea del timer invocará la función **timer_ISR** (localizada en main.c) para tratarla y luego borrará el flag utilizando el interfaz definido en **intcontroller.h**.
  - Si es una interrupción por la línea **EINT4567** invocará la función **button_ISR** (localizada en main.c) para tratarla y después borrará el flag de interrupción utilizando el interfaz definido en **intcontroller.h**.

La función main de la práctica 3 será por tanto:

```
int main(void)
{
	setup();
	while (1) {

	}
	return 0;
}
```

Como ayuda final, veremos qué valores debemos dar a los registros de configuración del *TIMER0* para que produzca interrupciones periódicas de periodo M segundos (en nuestro caso M=2). De acuerdo con la documentación del chip, la frecuencia con la que trabajan los timers depende de tres factores: la frecuencia del sistema (MCLK), el factor de pre-escalado (P) y el factor de división (D). Conocidos estos valores la frecuencia de funcionamiento sería:

$$F = \frac{MCLK}{(P+1) \cdot D}$$

(2.1)

Queremos que se produzcan interrupciones cada M segundos contando N ciclos de frecuencia F, con 1 <= N <= 65535 (ya que el contador es de 16 bits), es decir:

$$1/F \cdot N = M \qquad (2.2)$$

$$\frac{(P+1) \cdot D}{MCLK} \cdot N = M$$

(2.3)

$$P = \frac{M \cdot MCLK}{N \cdot D} - 1$$

(2.4)

Nos interesa que M · MCLK sea divisible entre N · D, con P <= 255, ya que tenemos sólo 8 bits para el factor de pre-escalado. Por lo tanto, tomaríamos N como el mayor divisor de M · MCLK/D, representable con 16 bits.

En la placa MCLK = $64 \cdot 10^6 = 2^{12} \cdot 5^6$. Tomando D = 8, tendríamos que N debe ser el mayor divisor de $M \cdot 2^9 \cdot 5^6$. Para M = 2 quedaría que N debe ser el mayor divisor de $2^{10} \cdot 5^6$, es decir: $N = 5^6 \cdot 2^2 = 62500$. Y entonces nos quedaría:

$$P = \frac{2 \cdot 64000000}{62500 \cdot 8} - 1 = 255$$

Por lo tanto, para un periodo de 2 segundos exactos podemos tomar el factor de división 8, con pre-escalado 255 e inicializar la cuenta con 62500. El valor del registro de comparación puede ser cualquier valor mayor que 0 y menor que el número de cuenta, en este caso, 62500.

Bibliografía

[arm] Arm architecture reference manual. Accesible en http://www.arm.com/miscPDFs/14128.pdf. Hay una copia en el campus virtual.

[TPGa] Christian Tenllado, Luis Piñuel, and José Ignacio Gómez. Introducción al entorno de desarrollo eclipse-arm.

[TPGb] Christian Tenllado, Luis Piñuel, and José Ignacio Gómez. Descripción del sistema de memoria y de entrada/salida en la placa S3CEV40.

[um-] S3c44b0x risc microprocessor product overview. Accesible en http://www.samsung.com/global/business/semiconductor/productInfo.do?fmly_id =229&partnum=S3C44B0. Hay una copia en el campus virtual.