Práctica 3
Estructura de Computadores

Entrada/salida mediante interrupciones vectorizadas

Índice general

3.1. Objetivos de la práctica

En esta práctica continuaremos nuestro estudio del sistema de entrada/salida del procesador ARM7TDMI, pero esta vez gestionando interrupciones. Los principales objetivos de la práctica son:

- Entender, conocer y saber manejar el sistema de E/S por interrupciones vectorizadas.
- Conocer y saber manejar dispositivos básicos como leds, pulsadores, displays, teclado y temporizadores.

Tras comprobar las limitaciones de la entrada/salida mediante espera de respuesta, en esta práctica analizaremos el sistema de E/S con interrupciones vectorizadas que nos proporciona el controlador de interrupciones del chip S3C44BOX. Además añadiremos dos dispositivos: el teclado y el temporizador.

3.2. Lecturas previas obligatorias

Es imprescindible leer los capítulos 1 a 8 del documento titulado *Sistema de memoria y de entrada/salida en la placa S3CEV40* ([TPG]) publicado a través del Campus Virtual. Sin leer y comprender completamente ese documento, es absolutamente imposible hacer esta práctica. Puedes consultar toda la documentación en [arm] y [um-].

3.3. E/S por interrupciones vectorizadas

Ejecutando el código de la práctica anterior habremos podido observar que en ocasiones tiene un comportamiento poco preciso, debido a que la espera software introducida hace que no siempre atendamos a los dispositivos.

En una esta práctica comenzaremos modificando el código de la práctica anterior para que funcione por interrupciones **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.

Posteriormente haremos una segunda modificación al código para añadir el teclado matricial, que también trataremos por interrupciones. Para comprender el funcionamiento del teclado debe leerse la sección 8 de [TPG].

3.4. Primera fase: paso a interrupciones vectorizadas y uso de temporizadores

Como se indicó anteriormente, el objetivo de esta primera parte es mantener exactamente el mismo comportamiento que en la práctica anterior, pero usando interrupciones vectorizadas.

Para que funcione el proyecto añadimos al proyecto cuatro 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 [TPG, um-].
- `timer.h` y `timer.c`: interfaz e implementación de funciones del módulo que maneja los temporizadores. El alumno deberá completar la implementación de todas las funciones de este módulo, siguiendo las indicaciones de los comentarios y la documentación de [TPG, um-].

Asimismo, en el fichero `main.c` deberemos realizar los siguientes cambios:

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

  - debemos registrar las rutinas `button_ISR` y `timer_ISR` como rutinas de tratamiento de interrupciones para el tratamiento de las señales `EINT4567` y `TIMER0` respectivamente.
  - debemos configurar el controlador de interrupciones para que active la línea IRQ en modo vectorizado, deje la línea FIQ enmascarada, configure las líneas `TIMER0` y `EINT4567` por la línea IRQ del procesador y las deje habilitadas.

- Función `loop`: estará vacía (dejaremos sólo el return).
- Función `timer_ISR`: será declarada como una rutina de tratamiento de interrupción por IRQ, para que el compilador la genere como RTI. Además, al final de la función se debe borrar el flag de interrupción utilizando el interfaz definido en `intcontroller.h`. 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`: al igual que la anterior será declarada como rutina de tratamiento de interrupciones por IRQ y deberá encargarse de borrar el flag de interrupción. 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).

3.4.1. Acerca de la configuración del temporizador

Como ayuda final, veremos qué valores podemos 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:

$$F = \frac{\text{MCLK}}{(P+1)\cdot D} \tag{3.1}$$

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:

$$1/F \cdot N = M \tag{3.2}$$

$$\frac{(P+1)\cdot D}{\text{MCLK}} \cdot N = M \tag{3.3}$$

$$P = \frac{M \cdot \text{MCLK}}{N \cdot D} - 1 \tag{3.4}$$

Nos interesa que $M \cdot \text{MCLK}$ sea divisible entre $N \cdot D$, 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 $M \cdot \text{MCLK}/D$, representable con 16 bits.

En la placa $\text{MCLK} = 64 \cdot 10^6 = 2^{12} \cdot 5^6$. 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 62500.

3.5. Segunda fase: incorporación del teclado matricial

Deberemos dar soporte al teclado matricial de la siguiente forma:

- Crearemos una función `keyboard_ISR` para el tratamiento de la interrupción por pulsación de tecla en el teclado matricial. Deberá declararse como RTI y registrarse adecuadamente para el tratamiento de la línea `EINT1`.
- En la función `setup` añadiremos a la configuración del controlador de interrupciones la configuración de la línea `EINT1` por IRQ y la habilitación de esta línea.

Al alumno se le proporcionarán:

- Dos nuevos ficheros `keyboard.h` y `keyboard.c` que definen e implementan el interfaz del módulo de teclado. El alumno deberá completar la función de escaneo definida en el fichero `keyboard.c`.
- El esqueleto de la función `keyboard_ISR` que el alumno deberá completar. Dicha función debe:
  - Esperar 20ms para eliminar los rebotes de presión
  - Escanear el teclado utilizando el interfaz definido en el fichero `keyboard.h`.
  - Cambiar la configuración del timer 0 para que la generación de interrupciones periódicas tenga un periodo distinto en función de la tecla pulsada:
    - Tecla 0: periodo de 2s, valor de cuenta 62500 y divisor 1/8
    - Tecla 1: periodo de 1s, valor de cuenta 31250 y divisor 1/8
    - Tecla 2: periodo de 0.5s, valor de cuenta 15625 y divisor 1/8
    - Tecla 3: periodo de 0.25s, valor de cuenta 15625 y divisor 1/4
    - Resto de teclas: no cambiaremos la configuración del timer
  - Esperar a que se deje de presionar la tecla leyendo el bit 1 del registro de datos del puerto G.
  - Esperar 20ms para eliminar rebotes de depresión.
  - Borrar el flag de interrupción por la línea `EINT1`

Bibliografía

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

[TPG] Christian Tenllado, Luis Piñuel, and José Ignacio Gómez. 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.