Gravity DS

Definición del proyecto (Tercera parte)

Esta parte se encarga de los temporizadores. El objetivo es configurar y hacer uso de dos de los temporizadores de la DS para controlar tiempos. El primer temporizador se configurará para contar los segundos transcurridos, y servirá para controlar los efectos del logotipo inicial y de la cuenta atrás. El segundo temporizador interrumpirá 15 veces por segundo con el fin de ajustar el sonido que se reproduce al coger una moneda.

Para esta tarea, se definirán las siguientes funciones:

Fichero temporizadores.c:

```
void prepararTemporizador(int id, int frecuencia) {}
```

Realiza los cálculos del valor *latch* para que se generen interrupciones con la frecuencia (interrupciones por segundo) indicada en el parámetro, siguiendo la fórmula explicada en la teoría. Establece el registro *TIMERX_DAT* con el valor calculado. Si fuera necesario, se modificaría el registro *TIMERX_CNT* con las condiciones que hicieran falta.

```
void iniciarTemporizador(int id) {}
```

Esta función hace que el temporizador *id* comience a contar estableciendo a 1 el bit 7 del registro *TIMERX_CNT*.

```
void pararTemporizador(int id) {}
```

El objetivo de esta función es detener la cuenta del temporizador *id*. Para ello, se modifica el bit 7 del registro *TIMERX_CNT*.

Nuestro juego hará uso de un contador de segundos. Esto se consigue preparando el temporizador 0 a una interrupción por segundo y activando las interrupciones del temporizador. Cada segundo se enviará una interrupción que la siguiente rutina se encargará de gestionar, sumando una unidad a una variable de tiempo.

```
void intTemporizador0() {} // Rutina de atención a las interrupciones del temporizador 0

int tiempo_juego = 0;
```

El otro temporizador se usará para aumentar la frecuencia y reducir el volumen del sonido que se oye al pulsar una moneda, creando un efecto sonoro al estilo Mario Bros. Como el sonido durará menos de 1 segundo, hay que ajustar el temporizador 1 para que interrumpa más veces por segundo (estimamos que harán falta unas 15 interrupciones por segundo). El ajuste del sonido se realizará en una función del fichero *sonido.c*.

```
void intTemporizador1() {} // Rutina de atención a las interrupciones del temporizador 1
```

Fichero temporizadores.h:

```
extern void prepararTemporizador(int id, int frecuencia);

extern void iniciarTemporizador(int id);

extern void pararTemporizador(int id);

extern void intTemporizador0();

extern int obtenerTiempo(); // Esta función devuelve el valor de la variable 'tiempo'

extern void resetearTiempo(); // Pone la variable 'tiempo' a cero

extern void intTemporizador1();
```

Casos de prueba:

Para comprobar el correcto funcionamiento de este apartado del proyecto, se desarrolla un pequeño programa que simplemente cuenta e imprime en pantalla el tiempo.

Se utiliza `consoleDemoInit()` para imprimir por pantalla con `printf()` de forma sencilla. Los pasos que da el programa son:

1. Preparar el temporizador 0 para que mande una interrupción por segundo.
2. Contar hasta 20 (el gestor de interrupciones se encargará de llamar a la rutina `intTemporizador0`).
3. Poner a cero el contador
4. Volver a contar hasta 10

Definición del proyecto (Cuarta parte)

La última parte es la encarga de gestionar las interrupciones que se producen para darles una respuesta conveniente. Se utiliza el *Interrupt Dispatcher* para leer la tabla de interrupciones y ejecutar la instrucción correspondiente.

Fichero interrupciones.c:

```
void rellenarTablaInt() {}
```

Está función se encargará de llamar a la función **irqSet** de *libnds* las veces que haga falta para especificar las rutinas de atención a cada tipo de interrupción. En nuestro caso, serán las funciones de atención a interrupciones de refresco vertical y temporizador. Se tendrá en cuenta el orden de llamadas a **irqSet** para definir las prioridades de interrupción. Esta misma función se encargará de llamar a las funciones de preparación de temporizadores.

En principio, no parecen necesarios más procedimientos en este apartado. Es el propio *Interrupt Dispatcher* el que se encarga de llamar las rutinas cuando se produce la interrupción y volver a la ejecución normal cuando termina.

Fichero interrupciones.h:

```
extern void rellenarTablaInt();
```

Casos de prueba:

Los casos de prueba indicados para este apartado serían el control de interrupciones para que se cumpla la transición de estados definida en el autómata del proyecto. Para comprobar que las interrupciones se comportan como queremos, hay que definir todos los estados, cerciorarse de que se llega a ellos como está previsto, y que no se producen transiciones anómalas.