Gravity DS

Definición del proyecto (Tercera parte)

Esta parte se encarga de los temporizadores. El objetivo es configurar y hacer uso de los cuatro temporizadores (o los que sean necesarios) de la DS para controlar el tiempo que lleva jugando el usuario desde el comienzo del nivel, y finalmente registrar el tiempo que ha aguantado vivo.

Para esta tarea, se definirán las siguientes funciones:

Fichero temporizadores.c:

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

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

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

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

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

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

Nuestro juego contará los segundos que pasan en la partida. Esto se consigue preparando un temporizador a una interrupción por segundo y activando las interrupciones del temporizador. Cada segundo se enviará una interrupción que la rutina de atención a las interrupciones se encargará de gestionar, sumando una unidad a una variable de tiempo.

```
void intTemporizador() {} // Rutina de atención a las interrupciones
```

Fichero temporizadores.h:

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

extern void iniciarTemporizador(int t_id);

extern void pararTemporizador(int t_id);

extern void intTemporizador();

extern int tiempo; // Esta variable controla el tiempo de juego
```

Esta variable será visible desde otras partes del programa, como por ejemplo, la que se encargue de controlar las puntuaciones.

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 un cronómetro.

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

1. Preparar un temporizador para que mande una interrupción por segundo.
2. Iniciar la cuenta
3. El gestor de interrupciones se encargará de incrementar la variable de tiempo y mostrar un mensaje.
4. Cuando llega a 1 minuto, parar la cuenta.

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 teclado y temporizador. Se tendrá en cuenta el orden de llamadas a **irqSet** para definir las prioridades de interrupción

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:

Un caso de prueba sencillo pero efectivo sería un pequeño programa que maneje las interrupciones de teclado y temporizador, e imprima un mensaje en pantalla con `printf()` cada vez que se produce una interrupción.