Memoria Técnica

Proyecto Hardware

Jaime Yoldi Vigueras 779057 José Marín Díez 778148

Enero 2021

Índice

1 Introducción

En este documento se presenta la memoria técnica correspondiente a las prácticas 2 y 3 de la asignatura Proyecto Hardware. A lo largo del documento se detalla el proceso de implementación del conocido juego Reversi. Partiendo del proyecto en lenguaje c previamente optimizado en la práctica 1 se realizan una serie de modificaciones y adaptaciones, añadiendo nuevas funcionalidades y permitiendo que este pueda ser ejecutado en un emulador de máquina real con sus periféricos, como pueden ser botones, leds, teclado, pantalla, etc.

Algunas partes del código serán desarrolladas en lenguaje `ARM` y otras en `c` para finalmente ser todo ello ejecutado con la ayuda de un emulador del procesador *ARM LPC2105*. El entorno de desarrollo empleado es *uVision IDE*.

2 Librerias

2.1 Botones

2.1.1 EINT0

2.1.2 EINT1

2.2 Temporizadores

2.2.1 TIMER0

2.2.2 TIMER1

2.3 GPIO

2.4 Cola de eventos

2.5 Gestión de los eventos

2.5.1 Máquina de estados

2.6 Comandos

2.7 Power Management

2.8 Modificaciones Reversi8

```
* Utilizacion del ARM_ARM
```

2.9 Modificaciones Startup

2.10 RTC

2.11 SWI

```
* TODO: Aumentar el valor de la tabla que nos dijo para que no desborde
```

2.12 UART0

2.13 RTC

Para medir el tiempo transcurrido durante la partida se hace uso de uno de los dos contadores con funcionalidades especificas del procesador *ARM LPC2105*, el *Real Time Clock* (RTC). Es necesario destacar que este contador, a diferencia de los otros no genera ninguna interrupción, ya que como se ha dicho su único propósito es proporcionar información sobre el tiempo transcurrido y consumiendo poca energía.

Este es iniciado al comienzo del juego y está en funcionamiento en todo momento incluso cuando el procesador está suspendido, en powerdown o idle. Antes de esto es necesario configurarlo para adaptarlo a la frecuencia a la que trabaja el procesador (60MHz), para ello se modifica el *Prescaler Integer register* y el *Prescaler Fraction register* de la siguiente manera.

```
PREINT = 0x726;
PREFRAC = 0x700;
```

Para que el RTC comience a contar es necesario habilitarlo, para ello se activa el bit 0 del *Clock Control Register*, además de poner a 0 la cuenta de minutos y segundos.

```
CCR = 0x01;
```

Para obtener el tiempo transcurrido se hacen funciones `RTC_leer_segundos` y `RTC_leer_minutos` que se encarga de leer del registro `CTIME0` los bits correspondientes y devolver el tiempo para cada caso.

2.14 Watchdog

Por otra parte, otra de las funcionalidades que tiene el juego es que se reinicia tras cierto tiempo de inactividad. Para ello, se utiliza el otro contador específico del procesador, el watchdog (WD).

Cuando el juego comienza, el Watchdog es iniciado especificándole el número de segundos a los que se quiere su reinicio. Para ello, se escribe en el registro *Watchdog Timer Constant* el numero de tics, en función de la frecuencia del procesador. Una vez realizado esto se habilita, se resetea su valor y se alimenta por primera vez para que comience a contar.

Cuando el WD se dispara se activa el segundo bit del *Watchdog Mode register* por lo que previamente se comprueba que no este ya disparado, limpiando el bit en caso afirmativo.

```
if( WDMOD & 0x04 )
    WDMOD &= ~0x04;

// Time out: Pclk*WDTC*4
WDTC  = (60000000 * sec) / 4;
WDMOD = 0x03;
feed_WT();
```

A partir de ese momento el temporizador se decrementa en cada pulsación de reloj, disparándose cuando su valor llegue a 0. La manera de evitar que se dispare es incrementar su tiempo de cuenta (alimentarlo) haciendo que comience de nuevo.

La manera de alimentar al Watchdog es mediante dos escrituras en el registro WDFEED. Para ello se ha creado la función `feed_WT`.

Es de gran importancia destacar que estas escrituras deben ser consecutivas, si no es así el correcto funcionamiento del programa se vera alterado. Por tanto es necesario asegurarse de que no va a llegar ninguna interrupción entre medio. Esto se consigue desactivando todo tipo de interrupción antes de las escrituras y activandolas de nuevo después. Para hacerlo se hará uso de las funciones `disable_isr_fiq` y `enable_isr_fiq` mencionadas anteriormente.

```
disable_isr_fiq();
WDFEED = 0xAA;
WDFEED = 0x55;
enable_isr_fiq();
```

Durante el transcurso del juego el encargado de alimentar el WD es el gestor de eventos, que lo hará cuando el jugador muestra algún tipo de actividad, ya sea la escritura de un nuevo comando o la pulsación de cualquiera de los dos botones. Si no se realiza ningún movimiento el contador continua decrementandose hasta llegar al final provocando que el procesador se resetee.

También es una manera de evitar que en caso de fallo el procesador se quede bloqueado, ya que si esta colgado el jugador no puede hacer nada, por lo que no será alimentado y también se reseteará.