

PRÁCTICA 5: ENTRADA/SALIDA

OBJETIVOS Y COMPETENCIAS A ADQUIRIR:

- Poner en práctica la gestión de periféricos en bajo nivel
- Consolidar conceptos relativos entrada y salida e interrupciones.

TRABAJO PREVIO:

Para la realización de esta práctica conviene estudiar y comprender los siguientes aspectos:

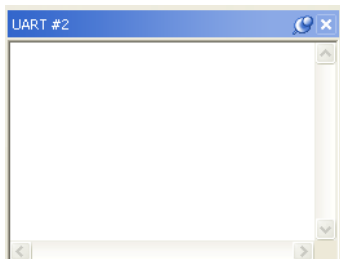
- Gestión de interrupciones en la arquitectura ARM v4.
- Funcionamiento del sistema de E/S del microcontrolador NXP LPC 2105 configurado para AOC1. Esta configuración básica es:
 - Procesador: Flags I=0, F=1, modo supervisor, pilas inicializadas (modos svc, irq).
 - Controlador de interrupciones (VIC): Todas las peticiones de interrupción quedan configuradas como IRQ vectorizadas manteniendo el orden de prioridades.
 - Funcionamiento del Timer 0 (100 interrupciones por segundo a través de la IRQ4 del VIC).
 - Conexión del teclado a través de la UART 1 (IRQ7 del VIC).
- Cualquier duda sobre el funcionamiento de este sistema de E/S se puede mirar en el manual de usuario del microcontrolador NXP LPC 2105 (disponible en moodle/hendrix).

En moodle/hendrix podéis descargar los ficheros necesarios para esta práctica. Consisten en un proyecto Keil configurado para la realización de prácticas de E/S. Al abrir el proyecto veréis que incluye dos ficheros fuente en ensamblador:

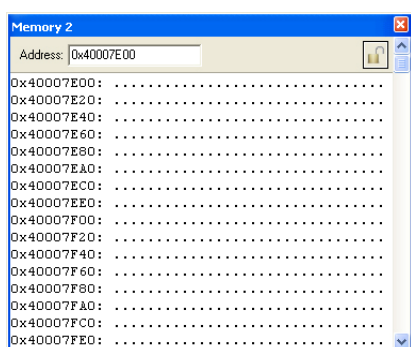
- **Startup.s** Contiene el código de inicialización del microcontrolador, VIC y periféricos. No debéis modificar nada de este fichero ni hace falta entenderlo.
- **prac5.s** Contiene las directivas necesarias para que la compilación y enlace con Startup.s funcione correctamente. Aquí debéis poner vuestro código de la práctica.
- **rand.s** Contiene el código para la generación de números pseudo-aleatorios.

Durante el desarrollo, depuración y ejecución de la práctica, se recomienda poner un punto de parada (breakpoint) en la primera instrucción de vuestro código en el fichero fuente prac5.s. De esta forma, con el comando (Run = F5) se ejecuta todo el código de inicialización de Startup.s seguido y el simulador se para en la primera instrucción de vuestro programa. A partir de ahí podéis depurar vuestro programa como queráis.


Para el correcto funcionamiento del teclado, es necesario añadir (en debug session) una nueva ventana (View->Serial Windows->UART#2). Esta ventana se corresponde con la UART1 (Universal Asynchronous Receiver/Transmitter) del LPC 2105 y hay que dejarla como ventana activa. Cualquier tecla pulsada en el teclado será registrada por la UART a través de esa ventana y ocurre lo siguiente:



- Se almacena en el puerto 0xE0010000 (equivalente a un registro de datos) el código ASCII de la letra correspondiente a la tecla pulsada (mayúsculas y minúsculas son distintas).
- Se genera una petición de interrupción por la IRQ7 del VIC. Esta petición se mantiene hasta que el programador lea el dato del puerto 0xE0010000.



Para observar las actividades del programa, se gestionará una zona de la memoria RAM (direcciones 0x40007E00 a 0x40007FFF) como si fuese una pantalla de texto (como en la práctica 4). Para ver el contenido de esa pantalla, conviene añadir (en debug session) una nueva vista de memoria (View->Memory Windows->Memory 2) con la geometría de la figura adjunta. Formato ASCII, 16 filas (numeradas de 0 a 15) y 32 columnas (numeradas de 0 a 31) a partir de la dirección 0x40007E00).

Para ver el estado del VIC debéis seleccionar el comando Peripherals -> Vectored Interrupt Controller (en debug session). Se recomienda no cambiar la configuración del VIC. Si por error se cambia la configuración del VIC, volver a ejecutar el código de Startup.s pulsando el botón de reset ().

Esta práctica incluye dos opciones una básica, cuya nota máxima alcanzable es un 7 y segunda opción más avanzada, que incorpora pequeñas variantes, algo más complejas y que permite alcanzar la máxima calificación.

OPCION 1: (NOTA MÁXIMA 7)

Abrir el proyecto “pract5” adjunto a la práctica. Diseñar, codificar e implementar un juego consistente en neutralizar al profesor de AOC1. La figura del profesor (carácter ‘#’, código ASCII 35, inicialmente en fila 1 columna 24) se mueve de izquierda a derecha de forma aleatoria siempre en la misma fila de la pantalla. Si el profesor rebasa un extremo de la pantalla, debe aparecer por el otro extremo, pero en la misma fila. Ritmo inicial de movimiento, 1 posición cada 0,08 segundos (según el reloj de simulación). El alumno se desplaza en un cañón (carácter ‘+’, código ASCII 43, inicialmente en fila 14 columna 8) de izquierda a derecha, también siempre en la misma fila, pulsando respectivamente las teclas ‘K’ (ASCII 75 ó 107) y ‘L’ (ASCII 76 ó 108). El cañón no puede pasar de un extremo a otro de la pantalla. En cualquier momento puede disparar pulsando la barra espaciadora (ASCII 32). La bala (carácter ‘*’ código ASCII 42) se mueve de abajo a arriba, siempre en la misma columna, desde la posición del cañón en el momento del disparo. Si al llegar a la fila 1, coincide con el profesor, el profesor ha sido neutralizado. Como mucho habrá una bala en la pantalla.

Para mejorar el juego, añadir tecla 'Q' (ASCII 81 o 113) para finalizar el programa, tecla '+' (ASCII 43) para duplicar la velocidad de movimiento (máximo 1 movimiento cada 0,01 segundos) y tecla '-' (ASCII 45) para dividir por dos la velocidad de movimiento (mínimo 1 movimiento cada 1,28 segundos).

La gestión del teclado y del timer debe realizarse con sincronización por interrupción y transferencia programada. Cualquier otro aspecto no especificado en la memoria, queda a libertad de elección del alumno.

El esquema del programa es el siguiente:

```

AREA datos, DATA
reloj      DCD 0          ;contador de centesimas de segundo
max        DCD 8          ;velocidad de movimiento (en centesimas s.)
cont       DCD 0          ;instante siguiente movimiento
dirx       DCB 0          ;direccion mov. caracter '+' (-1 izda.,0 stop,1 der.)
fin        DCB 0          ;indicador fin de programa (si vale 1)

AREA codigo, CODE
EXPORT inicio      ;etiqueta enlace con Startup.s

inicio
;programar @IRQ4 -> RSI_reloj
;programar @IRQ7 -> RSI_teclado
;activar IRQ4,IRQ7

bucle
;si toca movimiento
; calcular instante siguiente movimiento
; borrar '#' anterior
; calcular nueva posicion (direccion aleatoria) '#'
; dibujar nuevo '#'
; borrar '+' anterior
; calcular nueva posicion (sumando dirx) '+'
; dibujar nuevo '+'
; si hay disparo
; borrar '*' anterior
; calcular nueva posicion '*'
; dibujar nuevo '*'
; comprobar neutralizacion
;si fin=0 salto a bucle

;desactivar IRQ4,IRQ7
;desactivar RSI_reloj
;desactivar RSI_boton

bfin      b bfin

RSI_reloj  ;Rutina de servicio a la interrupcion IRQ4 (timer 0)
           ;Cada 0,01 s. llega una peticion de interrupcion

RSI_teclado ;Rutina de servicio a la interrupcion IRQ7 (teclado)
           ;al pulsar cada tecla llega peticion de interrupcion IRQ7
END

```

Una vez depurado vuestro programa, **para observar correctamente el movimiento del juego, activad la opción View -> Periodic Window Update (debug session)** y luego ejecutad todo el programa seguido (Run = F5). De esta forma se observa perfectamente el movimiento de las marcas de bits activos.

AYUDA:

Para la generación de números pseudo-aleatorios (imprescindibles cuando nos ponemos a jugar) podéis utilizar las subrutinas `rand` y `srand` del fichero fuente `rand.s`.

- **srand** sirve para inicializar la secuencia de números pseudo-aleatorios mediante una semilla (número natural de 32 bits). Esta subrutina **no devuelve ningún resultado** y tiene **un único parámetro** (la semilla) que se pasa **por valor en la pila**. Debéis invocar a esta SBR una única vez al comienzo del programa. Cambiando la semilla se obtiene una secuencia distinta de números pseudo-aleatorios.
- **rand** sirve para generar el siguiente número pseudo-aleatorio a partir del anterior. Esta subrutina **no tiene parámetros** y como **resultado** devuelve **en la pila** un **número pseudo-aleatorio de 31 bits**. Debéis invocar a esta SBR cada vez que necesitéis un número pseudo-aleatorio para vuestro programa.

OPCION 2: (NOTA MÁXIMA 10)

Añadir a la opción 1 al menos dos de las siguientes características:

- Permitir mas de una bala en movimiento a la vez.
- Posibilidad de que el profesor se defienda con disparos descendentes aleatorios. Añadir dos teclas para controlar la cantidad de disparos del profesor.
- Marcador de puntos. Crece con cada neutralización del profesor
- Cualquier otra característica que creáis conveniente para mejorar el juego.

EVALUACIÓN:

La evaluación de esta práctica se realizará durante la sesión de entrega de prácticas que se convocará en fecha próxima al examen de cada convocatoria (posible el mismo día del examen). Debe mostrarse el funcionamiento correcto de la práctica y responder a las preguntas que os hagan los profesores al respecto.