PRÁCTICA 5: ENTRADA/SALIDA

OBJETIVOS Y COMPETENCIAS A ADQUIRIR:

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

TRABAJO PREVIO:

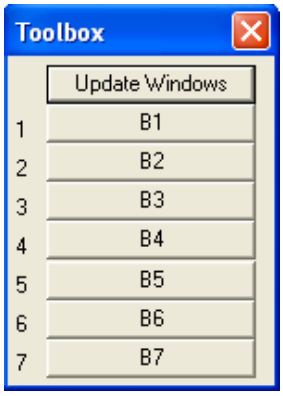
Para la realización de esta práctica conviene estudiar y comprender los siguientes aspectos:

- Gestión de excepciones tipo IRQ en la arquitectura ARM v4.
- Funcionamiento sistema 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).
- Funcionamiento del GPIO (General Purpose Input/Output), en concreto de los registros IOSET (0xE0028004) e IOCLR (0xE002800C). Lo podéis 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.

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 ejecutar instrucción por instrucción todo vuestro programa si hace falta.

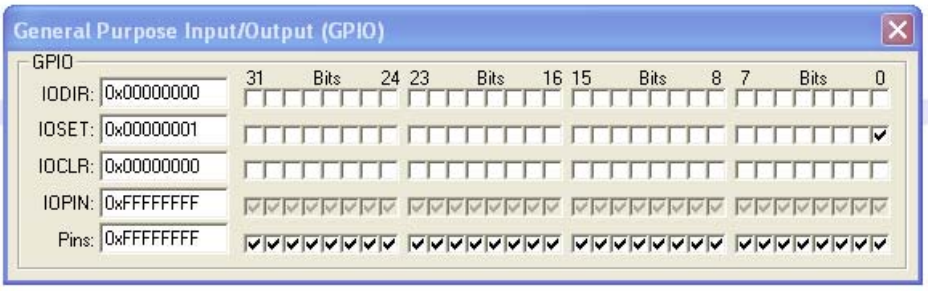


En el proyecto que se adjunta se han definido unos botones (ver figura adjunta) visibles cuando el simulador está en modo depuración (debug session). Esta ventana aparece y desaparece seleccionando el comando View->Toolbox Window ( ). Cada vez que se pulsa un botón numerado del 1 al 7 ocurre lo siguiente:

- Se almacena en la dirección 0xE001C008 (equivalente a un registro de datos) el código del botón pulsado (número de 1 a 7).
- Se genera una petición de interrupción por la IRQ1 del VIC. Esta petición se mantiene hasta que el programador la baje activando el bit 1 del registro VICSoftIntClear (0xFFFFF01C).
- El programador, después de leer el código del botón pulsado, debe escribir un 1 en el bit 2 de la dirección 0xE001C018 (equivalente a un registro de control) para indicar al controlador que se ha leído el código de botón pulsado.

Estos 7 botones constituyen una versión simplificada de un teclado. Se emplearán en la práctica para introducir comandos y controlar la ejecución del programa. Este periférico virtual habrá que gestionarlo con sincronización por interrupción.

Para observar el estado del GPIO debéis seleccionar el comando Peripherals -> GPIO (en debug session).



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 programa en ensamblador de ARMv4 que controle dos marcas en el registro IOSET (0xE0028004) del GPIO. La marca 1 comienza en el bit 0 y debe desplazarse hacia la izquierda una posición cada 0,16 segundos (según el reloj de simulación) y la marca 1 comienza en el bit 31 y se mueve hacia la derecha al mismo ritmo. Si una marca llega a un extremo del registro, debe dar la vuelta por el otro extremo continuando el movimiento a la misma velocidad y con la misma dirección. El programa debe terminar cuando se pulse el botón 1.

La función de los botones debe ser (se pueden cambiar las etiquetas en el fichero prac5.ini):

B1: Fin de programa
B2: Cambio dirección marca 1 (secuencia: izquierda, stop, derecha)
B3: + velocidad marca 1. Duplicar la velocidad (máximo 1 movimiento cada 0,01 segundos).
B4: - velocidad marca 1. Dividir por dos la velocidad (mínimo 1 movimiento cada 10,24 segundos).
B5, B6 y B7: Los mismos controles que B2, B3 y B4 respectivamente, pero para la marca 2.

La gestión de los botones y del timer debe realizarse con sincronización por interrupción y transferencia programada. El esquema del programa es el siguiente:

```
            AREA datos, DATA
reloj       DCD 0             ;contador de centesimas de segundo
max1        DCD 16            ;velocidad de mov. marca 1 (en centesimas s.)
max2        DCD 16            ;velocidad de mov. marca 2 (en centesimas s.)
cont1       DCD 0             ;instante siguiente movimiento marca 1
cont2       DCD 0             ;instante siguiente movimiento marca 2
dir1        DCB 0             ;direccion mov. marca 1 (-1 izda.,0 stop,1 der.)
dir2        DCB 0             ;direccion mov. marca 2 (-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 @IRQ1 -> RSI_boton
            ;programar @IRQ4 -> RSI_reloj
            ;activar IRQ1,IRQ4

bucle       ;para cada marca
            ;  si toca mover marca
            ;    calcular nueva posicion
            ;    calcular instante siguiente movimiento
            ;si cambia alguna marca
dibujar     ;  borrar IOSET
            ;  calcular nuevo IOSET
            ;  dibujar nuevo IOSET
            ;si fin=0 salto a bucle

            ;borrar IOSET
            ;desactivar IRQ1,IRQ4
            ;desactivar RSI_reloj
            ;desactivar RSI_boton

fin         b fin

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

RSI_boton   ;Rutina de servicio a la interrupcion IRQ1 (SW interrupt)
            ;al pulsar cada boton llega peticion de interrupcion IRQ1

            END
```

Una vez depurado vuestro programa, para observar correctamente el movimiento de las marcas de bits activos, 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.

OPCION 2: **(NOTA MÁXIMA 10)**

Igual que la opción 1 pero gestionando un número N (constante definida con EQU) de marcas. El programa empieza con 1 marca (en el bit 0 y movimiento a la izquierda cada 0,16 segundos) y se pueden añadir o borrar nuevas marcas con pulsaciones de botones. Cada nueva marca empieza como la primera. Si se borra una marca, por simplicidad, desaparece la última que se añadió. La función de los botones debe ser:

B1: Fin de programa
B2: Añadir marca
B3: Quitar marca
B4: Selección de marca. Cambia la marca objetivo de los comandos (secuencia cíclica: 1,2,...,k)
B5: Cambio dirección marca objetivo (secuencia: izquierda, stop, derecha)
B3: + velocidad. Duplicar la velocidad de la marca objetivo (max 1 mov. cada 0,01 segundos).
B4: - velocidad. Dividir por dos la velocidad de la marca objetivo (min 1 mov. cada 10,24 segundos).

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.