**ARQUITECTURA DE COMPUTADORES**
**BOLETÍN TEMA 3: ENTRADA/SALIDA**

## Analíticos

**Ejercicio 1.** Un periférico genera una interrupción cuando está preparado para realizar una operación. Además, dispone de un registro de estado cuyo bit menos significativo toma el valor 1 cuando está listo. A continuación, se muestra la RTI del periférico. Explique muy brevemente el error que contiene y su posible solución.

```
void rti (){
    ...
    while ((inb(STATUS_PORT_X) & 1)== 0);
    operacion_ES();
}
```

**Ejercicio 2.** Un computador con un procesador secuencial está conectado a un dispositivo de E/S que consta de dos sensores: se transfiere el valor tomado por cada sensor en cada petición de lectura. Teniendo en cuenta que:

- El dispositivo de E/S necesita un promedio de 109ns para procesar la petición.
- El envío de cada petición tarda 4ns y la transferencia del dato 2ns.
- El cambio de contexto necesita 10ns.
- La emisión de la señal de interrupción es inmediata.
- Suponiendo que se van a leer ambos sensores (por lo que requiere dos transferencias), calcule el porcentaje de tiempo que la CPU dedica a realizar las dos transferencias:
  - **a)** Utilizando E/S programada.
  - **b)** Utilizando E/S por Interrupciones considerando que la segunda petición de lectura se realiza dentro de la rutina de tratamiento de interrupción. Muestre los cálculos y describa su planteamiento ayudándose de una línea de tiempo.

**Ejercicio 3.** Un procesador de 1GHz de reloj con un CPI medio de 10 en el que cada operación de E/S requiere 10 instrucciones de CPU, está conectada a una impresora. La impresora imprime a una velocidad de 80 caracteres/segundo y no posee memoria interna. Calcule el tiempo total para imprimir 100 caracteres. Supondremos terminada la transferencia cuando se envíe el último carácter (sin esperar a que la impresora lo procese). Nota: el controlador de E/S posee un registro de datos de 8 bits y una velocidad de transferencia de 100Kb/segundo.

- **a)** Suponiendo que se utiliza E/S por polling con un bucle para la espera activa de 20 instrucciones. La espera activa es necesaria para esperar a que la impresora haya procesado el último carácter enviado. Suponga que, en código de la espera activa, la lectura del registro de estado correspondiente se realiza en la primera instrucción.
- **b)** Mediante interrupciones. Considerando que el tiempo desde que se activa la interrupción hasta que empieza la ejecución de la rutina de servicio de interrupción es de 500 ciclos.

**Ejercicio 4.** Tenemos un escáner que transmite datos a razón de 8000 Bytes/s. Se pide comparar qué método (DMA o interrupciones) sería más adecuado, es decir, consumiría menos ciclos del procesador sabiendo que:

- Por interrupciones, el tiempo gastado en cada transferencia de 4 Bytes es 15 ciclos, el procesador tarda un tiempo medio de 1 ciclo en cambiar de contexto y 4 ciclos en configurar una transferencia.
- Por DMA, iniciar cada transferencia de 1000 B cuesta 200 ciclos y el tratamiento de la interrupción al terminar la operación completa cuesta otros 150 ciclos. Además, el procesador dedica 175 ciclos en configurar el controlador de DMA y 75 ciclos al cambiar de contexto tras finalizar la operación.
- Indique qué mecanismo es más rápido y qué aceleración tiene respecto al otro.

## E/S mapeada (RISC-V)

**Ejercicio 5.** Dispone del siguiente código en el que se realiza un acceso a un dispositivo de E/S.

```
   addi x10,x0,0x100
a: lbu x12,0(x10)
   andi x13,x12,0x40
   bne x13,x0,a
   lbu x20,2(x10)
   slli x20,x20,6
   lbu x15,1(x10)
   andi x15,x15,0x3F
   or x15,x15,x20
   andi x15,x15,0xDF
   sb x15,1(x10)
```

***a)*** Describa el dispositivo al que se refiere y su funcionamiento. Para ello, utilice el recuadro de más abajo para detallar los registros internos que posee: dirección de cada registro, tipo de registro y los bits involucrados en la comunicación con una letra distintiva (y única) que lo identifique, deje en blanco el recuadro de los bits que no son de interés en el dispositivo. Se trata de que describa el comportamiento del dispositivo a través del código (no que indique el resultado final tras la ejecución).

Para determinar el funcionamiento del dispositivo, describa cómo actúan los bits identificados, indicando la letra identificativa y su comportamiento (Ej1: Cuando el bit X está '1'… Ej2: Para enviar un dato hay que…)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Tipo de registro y Dirección |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | **REG.** _____ **DIR 0x**_____ |
| | | | | | | | | **REG.** _____ **DIR 0x**_____ |
| | | | | | | | | **REG.** _____ **DIR 0x**_____ |

***b)*** Realice una versión del programa anterior en C mediante E/S separada por polling.

***c)*** Modifique el programa del enunciado para adecuarlo a un dispositivo idéntico cuyas señales de estado y control utilizan lógica inversa (las líneas activas en alta ahora son en baja y viceversa), las señales asociadas a datos deben mantenerse intactas. Indique solo las instrucciones que cambia en la zona habilitada para ello.

**Ejercicio 6.** Sea un **RISC-V** que se conecta a un dispositivo capaz de medir la temperatura y la humedad. Este dispositivo cuenta con los siguientes registros:

| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| **Reg. de control y estado (0xAC)** | X | **Leer** | X | **TH** | X | **Disp*** | X | X |

- **Leer**: Solicita la lectura en el sensor. El bit TH especificará qué magnitud debe leer.
- **TH:** Especifica si la petición es para leer la temperatura (0) o la humedad (1):
- **Disp*:** Indica si ya está disponible en el registro de datos la magnitud solicitada. **0** → Magnitud disponible.

**Reg. de datos (0xAD):** Registro de **16 bits** que guarda el valor solicitado. La temperatura puede ser negativa.

Realice un programa en **ensamblador** que guarde en ***x5*** la temperatura actual. **Incluya un comentario por línea**.

```
.text
     li x10,0xAC
     …
```

**Ejercicio 7.** Dispone de un procesador RISC-V con E/S mapeada en memoria y los tres siguientes registros de E/S:

| | ***7*** | ***6*** | ***5*** | ***4*** | ***3*** | ***2*** | ***1*** | ***0*** |
|---|---|---|---|---|---|---|---|---|
| ***Reg A (0x53)*** | *0* | *1* | *1* | *1* | *0* | *1* | *0* | *0* |
| ***Reg B (0x54)*** | *1* | *1* | *0* | *0* | *0* | *0* | *0* | *1* |
| ***Reg C (0x55)*** | *0* | *0* | *0* | *1* | *1* | *X* | *1* | *1* |

Partiendo del siguiente código de carga de direcciones:

```
.text
   li x10, 0x53   # Dirección de Reg A
   li x11, 0x54   # Dirección de Reg B
   li x12, 0x55   # Dirección de Reg C
```

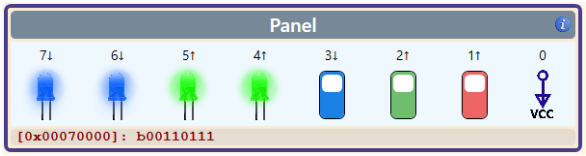
***a)*** Dado el siguiente trozo de código, indique (sobre el contenido de los registros anteriores), qué cambio habría.

```
lbu x20, 0(x10)
andi x20, x20, 0x10
srli x20, x20, 3
lbu x21, 0(x11)
or x21, x21, x20
sbu x21, 0(x11)
```

***b)*** Modifique el código anterior para que el cambio realizado afecte al bit 6 del registro C.

***c)*** Escriba un código en ensamblador de RISC-V para que conmute el valor del bit 2 del registro C: si es '0' debe pasar a valer '1'; y si es '1' debe pasar a valer '0'. Inicialmente, desconocemos su contenido (marcado como 'X' en el enunciado).

**Ejercicio 8.** Realice el código ensamblador que se solicita en cada apartado para un procesador RISC-V con E/S mapeada en memoria conectado a un dispositivo consistente en un panel con tres interruptores y cuatro leds. La flecha hacia arriba indica lógica positiva y hacia abajo lógica negativa. Por cada instrucción añada un breve comentario que describa la funcionalidad.




Suponiendo:

```
.text
li x6, 0x60000
```

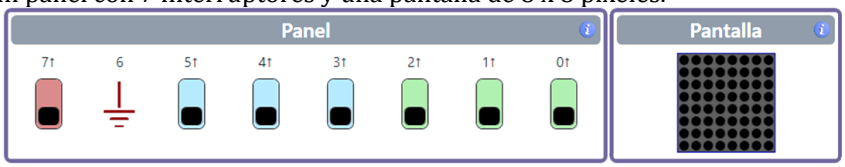
**a)** Espera activa mientras el interruptor conectado al pin 1 está desactivo**.**

**b)** Si el interruptor conectado al pin 2 está activo, apague los leds conectados a los pines 4 y 5.

**c)** Si interruptor conectado al pin 3 está activo, apague los leds conectados a los pines 6 y 7.

**d)** Encienda todos los leds.

**Ejercicio 9.** Se quiere implementar en ensamblador un juguete didáctico formado por un RISC-V conectado a dos dispositivos: un panel con 7 interruptores y una pantalla de 8 x 8 pixeles.




<u>Panel (0xFFFFF) – Dispositivo de entrada</u>:

- Bit 7: activo en alta. Borrado de la pantalla.
- Bits 5-3: identificador de columna en binario.
- Bits 2-0: identificador de fila en binario.

<u>Pantalla – Dispositivo de salida (formado por tres registros)</u>:

- Registro de Datos 1 (0x100000): configurar fila de la pantalla.
- Registro de Datos 2 (0x100001): configurar columna de la pantalla.
- Registro de Estado/Control (0x100002). Formado por:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | Clear | | Set* | | Busy | | |

  - Busy: activo en alta. Indica si está ocupado.
  - Clear: activo en alta. Borra la pantalla.
  - Set*: activo en baja. Activa el pixel especificado en los Registros de Datos 1 (fila) y 2 (columna).

Suponiendo que el registro '*x6*' posee el valor *0xFFFFF* y el registro '*x7*' el valor *0x100000*, se pide lo siguiente:

***a)*** Con el siguiente código, dibuje qué píxeles se activarían en la pantalla al finalizar su ejecución.

```
        addi x10, x0, 0
        addi x11, x0, 8
rep:    lbu x4, 2(x7)
        andi x4, x4, 0x04
        bne x4, x0, rep
        sb x10, 0(x7)
        sb x10, 1(x7)
        lbu x4, 2(x7)
        andi x4, x4, 0xEF
        sb x4, 2(x7)
        addi x10, x10, 1
        bne x10, x11, rep
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |

***b)*** Implemente el borrado de la pantalla. Para ello, consulte si está activo el bit correspondiente del panel y, en ese caso, borre la pantalla (teniendo en cuenta que no puede hacerlo si está ocupada).

***c)*** Implemente el código que lea la fila y columna del panel para, posteriormente, encenderlo en la pantalla.

## E/S separada (código C bajo Linux)

**Ejercicio 10.** Para un procesador que posee un sistema de E/S separada, escriba el código C que realice las tareas indicadas en cada apartado para el siguiente registro:

| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| **Dir 0x0F** | A (2 bits) | | B | C | D | E | F | G |

**a)** Modifique C a '1'.

**b)** Modifique E a '0'.

**c)** Consulte el valor de B. En caso de ser '0', escriba un '1' en G y, en caso de ser '1', escriba un '0' en F.

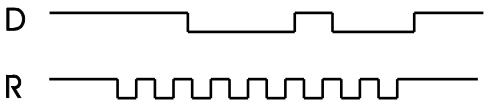
**d)** Ponga G a '1' si C vale '1' y F vale '0'.

**e)** Espere a que D valga '0' y, tras ello, copie el contenido de C a E.

**f)** Escriba el valor de la variable "unsigned char c" (que solo puede abarcar valores de 0 a 3), en A.

**Ejercicio 11.** Suponga un dispositivo E/S, con un único puerto de entrada de 8 bits, con el bit más significativo (MSB) normalmente a 1. Cuando baja a 0, indica que ha llegado un nuevo dato por los 7 bits restantes. Al leerlos, automáticamente el MSB vuelve a 1. Escriba una función que espere un dato (de 8 bits) y lo devuelva en una variable char.

**Ejercicio 12.** Se quiere construir un sistema de comunicación serie síncrona entre dos dispositivos, usando dos líneas: R y D. Los datos se reciben bit a bit por la línea D, con un nuevo bit por cada flanco de bajada del reloj R, y comenzando por el más significativo. Si no se reciben datos, la línea de reloj permanece a 1. Por ejemplo, el dato 11000100 (en binario):




Esta información es mandada por el dispositivo origen, mientras que el dispositivo destino se encarga de controlar los valores del reloj e ir leyendo de la línea de datos cuando sea oportuno. El receptor, dispone de un controlador de E/S que recoge el valor de estas dos señales de manera constante y lo guarda en un registro de datos (al guardar un nuevo valor, sustituye el valor anterior del registro anteriormente mencionado), al que se puede acceder mediante una lectura a la dirección 0x1000. En este registro de 8 bits, el bit 0 se encuentra conectado a la señal de reloj (R) y el bit 1 a la señal de datos (D).

Además, tiene un registro de control (accesible en la dirección 0x1001) mediante el que se controla la activación del dispositivo: El bit 0 habilita el dispositivo si está a uno, y el bit 1, habilita la generación de interrupciones). Si el dispositivo genera interrupciones, producirá una interrupción al procesador cuyo vector de interrupción es el 27).

**a)** Diseñe un programa que inicialice el periférico para la adquisición de datos por E/S por polling. El programa ha de recoger 1024 bytes y después, desactivar el dispositivo. Recuerde que, por cada byte, tendrá que hacer 8 lecturas de la señal D (una por bit), esperando los flancos de bajada de la señal R. Además, tendrá que concatenar estos bits mediante desplazamientos para obtener un byte. Cada vez que forme un byte, deberá almacenarlo en un vector de 1024 posiciones (unsigned char v[1024];).

**b)** Realice el apartado anterior usando interrupciones. Para ello, la interrupción se disparará ante un flanco de bajada de la señal R. Deberá hacer una única lectura de D y controlar el número de bits (de los 8 del byte) que lleva leídos, así como el total de 1024 bytes límite a leer.

**Ejercicio 13.** Sea un dispositivo para el control de nivel de agua que consta de 4 bombas de agua y 4 sensores: sensor inteligente que cambia a 1 cuando está lleno y cambia a 0 cuando está vació. Cada sensor y cada bomba está instalado en un depósito, de forma que se pueden controlar 4 depósitos. El dispositivo es manejado a través de los siguientes registros:

**Registro de Control (puerto 0x30):**

| ***Dir.*** | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---|---|---|---|---|---|---|---|---|
| ***0x30*** | ***B1*** | ***B0*** | ***ACTIVAR_B*** | ***X*** | ***X*** | ***S1*** | ***S0*** | ***LEER_S**** |

- **LEER_S***: Solicita la lectura (0) del sensor de agua indicado en los bits S1-S0.
- **S1-S0**: Sensor de agua a leer.
- **B1-B0**: Bomba a activar.
- **ACTIVAR_B**: Activa (1) o desactiva (0) la bomba de agua indicada en los bits B1-B0.

**Registro de Estado (puerto 0x31):**

| ***Dir.*** | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---|---|---|---|---|---|---|---|---|
| ***0x31*** | ***NIVEL*** | ***X*** | ***X*** | ***X*** | ***X*** | ***LEÍDO**** | ***X*** | ***X*** |

- **LEÍDO***: Se activa (0) cuando finaliza la petición de lectura.
- **NIVEL**: Una vez finalizada la petición de lectura del sensor, indicará con 1 si el último estado fue lleno, y 0 si el último estado fue vacío.

***a)*** Usando E/S por polling, realice en C una función que reciba como parámetro el depósito (posibles valores: 0, 1, 2, 3) del que se quiere leer el sensor y devuelva 1 si el último estado fue lleno y 0 en caso contrario.

***b)*** Utilizando E/S por polling realice en C la función principal que verifique continuamente si algún depósito está vacío usando la función anterior. En dicho caso, debe activar la bomba de dicho depósito y esperar hasta que se detecte lleno. En el momento que se detecte lleno, debe apagar la bomba y continuar con el control del resto de depósitos.

**Ejercicio 14.** Dispone de un microcontrolador que gobierna un sistema de recolección de datos proveniente de varios sensores. Consta de tres sensores: presión, inclinación y altitud. El microcontrolador posee un controlador de entrada/salida separada con las siguientes características:

**Registro de datos (puerto 0x30)**

**Registro de control (puerto 0x31):**

| *Dir.* | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---|---|---|---|---|---|---|---|---|
| *0x31* | ***ON*** | ***INT**** | ***S1*** | ***S0*** | ***LEER_S**** | ***X*** | ***X*** | ***X**** |

- **ON**: poner a 1 para encender el dispositivo
- **INT***: poner a 0 para activar modo por interrupciones, poner a 1 para activar modo polling
- **S1-S0**: indicar sensor a leer. 0: presión; 1: inclinación; 2: altitud
- **LEER_S***: poner a 0 para solicitar lectura del sensor indicado previamente en los bits S1-S0.

**Registro de Estado (puerto 0x32):**

| *Dir.* | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---|---|---|---|---|---|---|---|---|
| *0x32* | ***X*** | ***X*** | ***X*** | ***X**** | ***LEÍDO**** | ***ON*** | ***X*** | ***X*** |

- **BUSY***: vale 0 si el dispositivo está ocupado
- **LEÍDO***: Se activa (0) cuando finaliza la petición de lectura y hay un dato que leer en el registro de datos
- **ON**: vale 1 si el dispositivo está encendido y 0 en caso contrario

**a)** Implemente la función de configuración inicial del sistema. Dicha función recibe como parámetro un valor entero que indica si se configura en modo polling (1) o en modo por interrupciones. En caso de habilitación por interrupciones, se debe registrar con el vector de interrupción 0xA. Antes de empezar con la configuración, debe comprobar si está encendido, en caso contrario, debe encenderlo y esperar a que lo esté.

**b)** Mediante polling, realice una función que reciba como parámetro el sensor que se desea leer (posibles valores: 0, 1, 2) y devuelva el valor de dicho sensor. Para ello, debe establecer en los bits S1-S0 el sensor que se desea leer, activar LEER_S*, esperar la activación de LEÍDO* y leer el dato del registro de datos.

**Ejercicio 15.** Se quiere desarrollar un sistema para la medición de monóxido de carbono (CO) y activación de alarma en el caso de alcanzar un valor perjudicial para la salud. Para ello, dispone de un dispositivo, que consta de 4 sensores de CO (8 bits de resolución) y 4 sirenas, conectado a un microcontrolador de bajo coste. El dispositivo es capaz de emitir interrupciones para detectar un nivel de CO nocivo (ALARM) y para indicar la finalización de una operación de lectura. El sistema posee un módulo de entrada/salida separada con las siguientes características:

**Registro de control (puerto 0x30):**

| *Dir.* | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---|---|---|---|---|---|---|---|---|
| *0x30* | ***ON*** | ***INT**** | ***C_ALARM**** | ***SIRENA*** | ***S1*** | ***S0*** | ***LEER_S**** | ***X*** |

- ***ON:*** poner a 1 para encender el dispositivo.
- ***INT*:*** poner a 0 para activar modo por interrupciones, poner a 1 para activar modo polling.
- ***LEER_S****: Solicita la lectura (0) del sensor de nivel de CO indicado en S1-S0.
- **S1-S0**: Indica sensor a leer.
- **C_ALARM**: Configura la activación automática de la alarma (1).
- **SIRENA**: Activa (1) la sirena de la habitación indicada por los bits S1 y S0.

**Registro de Estado (puerto 0x31):**

| *Dir.* | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---|---|---|---|---|---|---|---|---|
| *0x31* | ***ALARM*** | ***S_ALARM1*** | ***S_ALARM0*** | ***X*** | ***X*** | ***LEÍDO**** | **DESACT*** | ***LISTO*** |

- **DESACT***: Se activa (0) cuando el operario pulsa el botón para desactivar la alarma.

- **LEÍDO***: Se activa (0) cuando finaliza la petición de lectura. En ese momento, está disponible el valor medido de CO en el registro de datos.
- **LISTO**: vale 1 si el dispositivo está encendido y 0 en caso contrario
- Una vez leído el registro, se desactivan.
- **ALARM:** Notifica (1) que ha saltado la alarma por exceso de CO en la habitación indicada por ***S_ALARM1*** y ***S_ALARM0.***

**Registro de Datos (puerto 0x32)**

***a)*** Implemente la función de configuración inicial del sistema. Dicha función recibe como parámetro un valor entero que indica si se configura en modo polling (1) o en modo por interrupciones (0). En caso de habilitación por interrupciones, se debe registrar con el vector de interrupción 0xA. Antes de empezar con la configuración, debe comprobar si está encendido, en caso contrario, debe encenderlo y esperar a que lo esté listo.

***b)*** Considerando el dispositivo configurado, realice en C mediante E/S por polling una función que reciba por parámetros el número del sensor cuyo valor de CO queremos leer y lo devuelva.

***c)*** Implemente la rutina de tratamiento de la interrupción que active la sirena cuando se active ***ALARM*** del registro de estado. La rutina sólo atenderá la activación de la alarma. Cuando salte la alarma deberá activar la sirena vinculada al sensor que ha provocado la alarma.

**Ejercicio 16.** Disponemos de un sistema de control automático de una silla de ruedas en base al movimiento de la cabeza de un usuario tetrapléjico. Para ello, realizaremos el control del sistema a través de dos registros de E/S comunicados con el controlador de la silla a través de E/S separada:

| Mov. cabeza | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Dir. 0xA** | *Lee* | *Disponible* | *X* | *X* | *Mov. Lateral* | | *Mov. Inclinación* | |

- (Datos) Mov. Lateral: Dato del movimiento lateral (derecha e izquierda) de la cabeza
- (Datos) Mov. Inclinación: Dato del movimiento de inclinación (adelante y atrás) de la cabeza.
- (Control) Lee: activo en alta. Activar para solicitar lectura de datos (tanto *Lateral* como *Inclinación*).
- (Estado) Disponible: activo en alta. Aviso de dato disponible para lectura.

| Motores Silla | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Dir. 0xB** | *COMANDO* | | | | | | *Ocupado* | *Activar** |

- (Estado) Ocupado: activo en alta. Aviso de motores ocupados.
- (Control) Activar*: activo en baja. Activar para solicitar ejecución de comando.
- (Datos) Comando: Dato que proporciona la orden de movimiento del motor de la silla de ruedas.

Se pide realizar un programa que, mediante E/S por polling, lea continuamente los valores de los movimientos *lateral* e *inclinación* del usuario y active el comando oportuno en el motor de la silla. Para ello, tenga en cuenta:

- Para leer los datos del usuario, se debe solicitar la lectura de los mismos mediante la señal de control correspondiente y esperar a que estén disponibles (señal de estado) antes de leer ambos.
- Para obtener el comando a aportar a los motores, una vez tenemos movimiento lateral e inclinación, debe usar la función ya implementada:

```
unsigned char comando (unsigned char lateral, unsigned char inclinacion)
```

- Para programar un movimiento en los motores de la silla solo podrá hacerlo si el controlador no está ocupado; y, una vez colocado el comando (obtenido de la función anterior), deberá activarlo mediante la señal de control correspondiente.

**Ejercicio 17.** Queremos programar el funcionamiento de los botones de un controlador de persiana eléctrica mediante E/S por interrupciones. Para ello, disponemos del siguiente registro de E/S ligado al controlador de la persiana:

| Persiana | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Dir. 0x05** | *X* | *Pul_Arriba* | *Pul_Abajo* | *X* | *Fin_Arriba** | *Fin_Abajo** | *Mov_Arriba* | *Mov_Abajo* |

- (Estado) Pul Arriba: activo en alta. Avisa que el usuario ha accionado el pulsador de recogida de persiana.
- (Estado) Pul Abajo: activo en alta. Avisa que el usuario ha accionado el pulsador de extensión de persiana.
- (Estado) Fin Arriba*: activo en baja. Avisa si la persiana ha llegado a su punto máximo de recogida.
- (Estado) Fin Abajo*: activo en baja. Avisa si la persiana ha llegado a su punto máximo de extensión.
- (Control) Mov Arriba: activo en alta. 1: iniciar el movimiento de recogida de persiana, 0: parar movimiento. También sirve para conocer si la persiana está en movimiento de recogida.
- (Control) Mov Abajo: activo en alta. 1: iniciar el movimiento de extensión de persiana, 0: parar movimiento. También sirve para conocer si la persiana está en movimiento de extensión.

El controlador provocará una interrupción si se activa cualquiera de sus cuatro bits de estado. Se actuará de la siguiente forma en base al evento recibido:

- Si el usuario ha pulsado el botón de recogida/extensión de persiana, deberemos activar el movimiento de recogida/extensión siempre y cuando no se encuentre la persiana en su posición máxima de recogida/extensión.
- Si la persiana llega al punto máximo de recogida/extensión, deberemos parar el movimiento de recogida/extensión.
- Si la persiana se encuentra en movimiento y el usuario realiza una pulsación en el mismo sentido que el movimiento actual (recogida o extensión), se parará la persiana en la posición en la que se encuentre.
- La parada de la persiana cuando llega a alguno de sus puntos máximos es prioritaria frente al resto de eventos.

Se pide programar la rutina de tratamiento de la interrupción (o rutina de servicio) para atender cualquiera de las fuentes de interrupción descritas, implementando el comportamiento indicado anteriormente.

**Ejercicio 18.** Como ingeniero del ejército de aviación, le encargan diseñar un sistema de actuación automática para posibles ataques sobre los aviones de última generación. Para ello, dispone de un conjunto de sensores que detectan automáticamente todos los peligros cercanos al avión y a cuyas señales de estado puede acceder a través del siguiente registro:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Dir. 0x10** | *MotorIz* | *MotorDe* | *Fuselaje** | *X* | *PresenciaIz* | *PresenciaDe* | *PresenciaTr* | *X* |

- Fuselaje*: activo en baja. Se activa si ha habido alguna perforación en el fuselaje.
- PresenciaIz: activo en alta. Proyectil detectado directo al avión en la cercanía del ala izquierda.
- PresenciaDe: activo en alta. Proyectil detectado directo al avión en la cercanía del ala derecha.
- PresenciaTr: activo en alta. Proyectil detectado directo al avión en la cercanía de la cola.
- MotorIz: activo en alta. Fallo del motor izquierdo.
- MotorDe: activo en alta. Fallo del motor derecho.

De igual forma, para intervenir sobre el sistema de actuación, puede hacerlo a través del siguiente registro:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Dir. 0x11** | *Expulsión** | *Potencia MotorIz* | | *Potencia MotorDe* | | *Contramedidas** | *AlerónIz* | *AlerónDe* |

- Expulsión*: activo en baja. Activar para expulsión inmediata del piloto.
- Potencia MotorIz: lectura/escritura. Consultar potencia actual del motor (de 0 a 3). También permite modificarla.
- Potencia MotorDe: lectura/escritura. Consultar potencia actual del motor (de 0 a 3). También permite modificarla.
- Contramedidas*: activar para lanzar contramedidas.
- AlerónIz: activo en alta. Activar para desplegar el flap del ala izquierda.
- AlerónDe: activo en alta. Activar para desplegar el flap del ala derecha.

*Nota: Todas las señales de control y estado, una vez se realiza la acción vuelven automáticamente a su valor inicial.*

La activación del sistema de actuación automática se realizará por interrupciones en base a los estados de peligro detectados. Se actuará de la siguiente forma en base al evento recibido:

1. Si detecta alguna perforación del fuselaje, el sistema debe expulsar automáticamente al piloto.
2. Si detecta un fallo en uno de los motores, deberá aumentar su potencia (en 1); salvo si su potencia actual es cero, en cuyo caso debe expulsar automáticamente al piloto.
3. Si detecta la cercanía de un proyectil en la parte trasera, deberá activar las contramedidas.
4. Si detecta la cercanía de un proyectil a alguna de las alas, debe aumentar la potencia del motor opuesto (al máximo) y activar el flap de dicha ala (con el fin de realizar una acrobacia para evitar el proyectil).

Se pide programar la rutina de tratamiento de la interrupción para atender cualquiera de las fuentes de interrupción descritas, implementando el comportamiento indicado anteriormente.

**Ejercicio 19.** Se quiere programar un microcontrolador para el control del motor de un sillón articulado mediante E/S separada. El controlador de E/S del motor dispone de los siguientes registros:

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Dir. 0x0A** | *Enable* | *Ready* | *Int** | *X* | *Bot_Subida** | *Bot_Bajada** | *Mov_Arriba* | *Mov_Abajo* |
| **Dir. 0X0B** | *Posición en Grados* | | | | | | | |

- (Control) Enable: Activar para habilitar el dispositivo
- (Estado) Ready: Indica que el dispositivo está listo para funcionar.
- (Control) Int*: Configura el dispositivo para actuar por interrupciones (0).
- (Estado) Bot_Subida*: Avisa que el usuario ha accionado el botón de subida (0).
- (Estado) Bot_Bajada*: Avisa que el usuario ha accionado el botón de bajada (0).

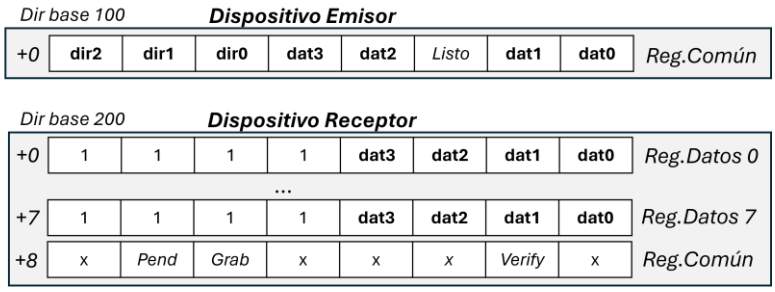
- (Control) Mov Arriba: Activar para iniciar el movimiento de subida y desactivar para parar dicho movimiento. También sirve para conocer si el sillón está en movimiento de subida.
- (Control) Mov Abajo: Activar para iniciar el movimiento de bajada y desactivar para parar dicho movimiento. También sirve para conocer si el sillón está en movimiento de bajada.
- (Dato) Posición en Grados: Contiene la posición en que se encuentra el motor. Rango [0, 45] grados.

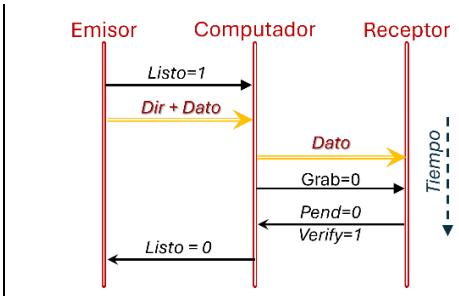
***a)*** Realice la función de configuración para el funcionamiento por interrupciones. Para configurar el dispositivo, previamente se debe comprobar si está listo, y en caso contrario, se debe habilitar y esperar a que esté listo. Para configurar el computador, se debe establecer la rutina de servicio de interrupción y habilitar la interrupción 0x7. Si se produce error devolverá 0, en caso contrario 1.

***b)*** Realice la rutina de servicio de la interrupción para atender a la interrupción que provoca el controlador si se pulsa alguno de los botones. Se debe actuar de la siguiente forma:

I. Si el sillón está parado, deberemos activar el movimiento correspondiente al botón pulsado, siempre y cuando no se encuentre el sillón en su posición máxima de subida/bajada (0 y 45 grados respectivamente). La posición está disponible en el segundo registro del controlador.
II. Si el sillón se encuentra en movimiento, se debe para el movimiento.
III. No es necesario que implemente la parada del motor al llegar a su posición máxima de subida/bajada porque el dispositivo la realiza de forma automática.

**Ejercicio 20.** Sea un computador conectado a dos dispositivos, uno Emisor y otro Receptor. El computador recibirá del dispositivo Emisor una dirección y un dato, y el computador deberá reenviar el dato al registro del dispositivo Receptor cuya dirección coincide con la proporcionada por el Emisor. Por ejemplo, si recibe del Emisor la dirección 4, enviará el dato recibido al registro de datos 4 del Receptor. El protocolo de comunicación para transferir un dato es el siguiente:



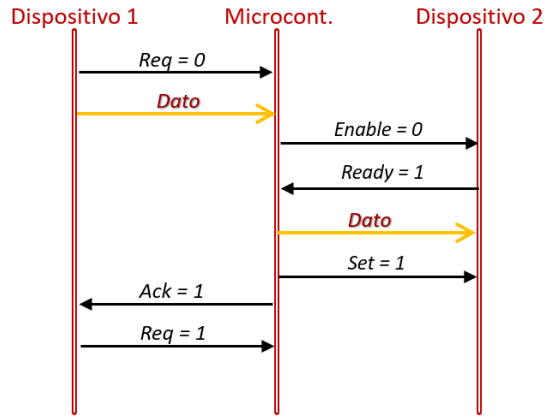




Tenga en cuenta que el *Receptor* confirma la grabación del Dato estableciendo *Pend*=0 y *Verify*=1 simultáneamente. Implemente en el *Computador* un programa en C mediante E/S separada que retransmita continuamente los datos recibidos del emisor al receptor según el cronograma anterior.

**Ejercicio 21.** Sea un sistema compuesto por dos dispositivos conectados a un microcontrolador, que se quiere programar para recibir datos del dispositivo 1 y trasmitirlo al dispositivo 2 según los siguientes criterios:

- **si el dato recibido es par, se debe transmitir el valor recibido sin modificar**
- **si el dato recibido es impar, se debe transmitir el valor recibido incrementado en 1.**

Se define el comportamiento detallado en el siguiente cronograma, y en la descripción de los registros de los dispositivos:




**Registro Dispositivo 1 (0x19):**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| *Req** | *Ack* | *D3* | *D2* | *D1* | *D0* | *X* | *X* |

**Req***: Indica que hay un nuevo dato disponible en D3-D0.
**Ack**: Notifica al dispositivo que el dato ha sido procesado.
**D3-D0**: Dato.

**Registro Dispositivo 2 (0x2F):**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| *Enable** | *Ready* | *D4* | *D3* | *D2* | *D1* | *D0* | *Set* |

**Enable***: Activar para habilitar el dispositivo.
**Ready**: Indica que el dispositivo está listo para recibir.
**D4-D0**: Dato.
**Set**: Notifica al dispositivo que se ha enviado un comando.

Se pide realizar un programa que, mediante E/S por polling, realice de forma repetitiva: la recepción del dispositivo 1 y transmisión del dato al dispositivo 2, modificado o no según se indica en el párrafo anterior (resaltado en negrita).

**Ejercicio 22.** Poseemos un controlador de E/S que se encuentra conectado a un dispositivo meteorológico, que controla presión y humedad del ambiente. El controlador de E/S posee tres registros a los que poder acceder, cada uno de ellos de un byte de tamaño, situados en las posiciones 0x30 (control), 0x31 (estado) y 0x32 (datos) del mapa de E/S del sistema (E/S aislada en nuestro computador). El dispositivo meteorológico será interrogado e irá proporcionando la información correspondiente: nos puede proporcionar la información utilizando datos de simple precisión (32 bits) o de doble (64 bits), dependiendo de la configuración que le aportemos al inicio. Aportará información byte a byte (registro de datos de un byte) y tendremos, a posteriori, que ir unificando estas informaciones.

**Registro de control**:

| Modo[1] | Modo[0] | X | RESET* | REQ* | PRECISION | VEL[1] | VEL[0] |
|---|---|---|---|---|---|---|---|

- **Modo [1-0]:** indica el modo de funcionamiento, pudiendo ser modo por petición (00), por ráfaga (01) o por interrupciones periódicas (10). El modo por petición solo nos proporcionará un dato cuando se lo indiquemos mediante la señal REQ*. El modo ráfaga irá proporcionándonos datos con una cierta cadencia (utilizado para no ir haciendo peticiones en el modo E/S por polling). En el último modo el controlador se configura con interrupciones: al activarlas, la interrupción queda asociada a la activación de la señal DATA* del registro de estado, que se activa cada vez que hay un byte disponible (ver descripción posterior), y a la activación de la señal ERROR*.
- **REQ*:** Petición de dato. En modo "por petición", habrá que activarlo para recibir cada uno de los bytes; en modo "por interrupciones", no tiene uso, pues el dispositivo, una vez encendido, comenzará a colocar datos e indicarlo en DATA* (registro de estado).
- **RESET***: Enciende el dispositivo si no lo estuviese o lo resetea en caso de error.
- **PRECISION:** indica si los datos aportados van a ser de 32 bits (0) o de 64 (1).
- **VEL [1-0]:** Velocidad de la ráfaga en el modo "01". En otro modo no es utilizado.

**Registro de estado:**

| X | X | X | DATA* | ACK* | ON | BUSY* | ERROR* |
|---|---|---|---|---|---|---|---|

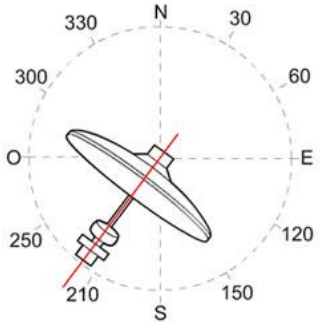
- **DATA*:** el periférico activa esta señal cuando ha colocado un byte disponible en el registro de datos. Posteriormente, podremos leerlo. Se activará tras poner cada byte (no únicamente con el primer byte de cada palabra).
- **ACK*:** confirmación por parte del periférico de recepción de una petición de dato (solo usado en modo "por petición"). La comunicación bajo demanda funciona entonces de la siguiente forma: el usuario activa REQ* en el registro de control, el periférico activa ACK* en el registro de estado al recibir la petición, el periférico coloca el dato en el registro de datos y, por último, activa la señal de DATA* para indicar que el dato está disponible. Esta comunicación se usará en el control bajo demanda, únicamente para el primer byte (los restantes los irá proporcionando automáticamente tras cada lectura, activando en cada caso la señal de DATA*). En modo "por interrupciones" esta comunicación entre REQ* y ACK* no es necesaria.
- **ON:** indica si está encendido.
- **BUSY*:** dispositivo ocupado.
- **ERROR*:** error de dispositivo. En ese caso resetearlo.

A la hora de solicitar los datos, el dispositivo nos devuelve en primera instancia la temperatura y, cuando termina con ella, comienza el envío de la humedad; así pues, no es necesario solicitar cada uno por separado. Se pide:

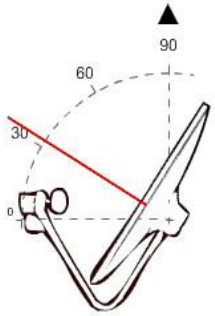
**a)** Realizar un control mediante E/S por polling para ir recibiendo, de forma periódica información en tiempo real del estado de los sensores (modo 00), utilizando para ello datos de simple precisión. Recuerde tener en cuenta el mecanismo de comunicación asíncrono que se ha explicado mediante el uso de REQ*, ACK* y DATA*. Tras recoger los datos, se almacenarán en un vector las temperaturas y en otro las humedades. Estos vectores tendrán un tamaño máximo de 1024 posiciones y, en caso de llenarse, se comenzará a escribir nuevamente por el principio.

**b)** Realizar un control por interrupciones (modo 10) del dispositivo creando, para ello, la rutina de inicialización (suponemos que se le asigna el vector de interrupciones 0xFF) y la rutina de interrupción del mismo. Esta rutina, tal como se comentó anteriormente, está asociada a la activación de DATA* y ERROR*. Así pues, deberá discriminar la fuente de interrupción para cada caso. En caso de error, se reinicia el dispositivo y, en caso de dato, se leerá el byte correspondiente (recuerde que una lectura de temperatura y humedad conlleva, al menos, 8 bytes; y en la rutina de interrupción leerá únicamente uno).

**Ejercicio 23.** Se desea controlar el giro de una antena parabólica mediante un computador. La antena dispone de 2 motores encargados de orientarla. El motor H controla el azimut (componente horizontal) y el motor V la Elevación (componente vertical).




*Azimut (motor H)*




*Elevación (motor V)*

Tales motores están controlados por un dispositivo que se encuentra conectado al computador. El computador se comunica con el dispositivo mediante E/S separada (aislada) y sólo puede utilizar E/S por polling (sondeo). El dispositivo contiene los siguientes registros:

**Registro de control de 8 bits (dirección de E/S: 0x350)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Función** | Dirección Motor H | Habilita Motor H | Home | Dirección Motor V | Habilita Motor V | Siempre a 0 | | |

- **Habilita Motor H o V:** con '1' el motor correspondiente gira en la dirección especificada por *Dirección Motor*. Con '0' el motor se detiene.
- **Dirección Motor V:** con '1' el giro del motor V hace aumentar el ángulo de elevación y con '0' lo contrario.
- **Dirección Motor H:** con '1' el giro del motor H hace rotar la antena en el sentido de las agujas del reloj (aumenta los grados del azimut) con '0' lo contrario.
- **Home:** Mueve los motores H y V hasta situarse en la posición inicial, esto es, elevación y azimut a 0 grados.

**Registro de estado de 16 bits (dirección de E/S: 0x351)**

| Bit | 15 14 13 12 11 10 9 8 7 | 6 5 4 3 2 1 0 |
|---|---|---|
| **Función** | Azimut actual en grados (valores posibles: de 0 a 359) | Elevación actual en grados (valores posibles: de 0 a 90) |

Para orientar la antena a un determinado satélite tendrá que activar cada motor y hacerlos girar en el sentido adecuado hasta alcanzar el satélite. Cuando alcance la elevación requerida deberá apagar el motor V inmediatamente, y de igual forma, deberá apagar el motor H cuando alcance el azimut requerido. Podrá verificar si ha alcanzado el ángulo requerido en ambos casos a través del registro de estado.

**Nota:** Recuerde que hay instrucciones de E/S para accesos de 8 bits (*inb* y *outb*) e instrucciones de E/S para accesos de 16 bits (*in* y *out*).

**a)** Realice una función en C que sitúe la antena en la posición inicial (Home). La función no debe terminar hasta que la antena no se encuentre situada en dicha posición.

*void orienta_a_Home()*

**b)** Realice una función en C que mueva el motor V hasta alcanzar la elevación en grados indicada como parámetro de la función (debe tener en cuenta la dirección hacia la que girar)

*void orienta_Elevacion (unsigned char elevacion)*

**Ejercicio 24.** Se dispone de un teclado con los siguientes registros de 8 bits:

**Registro de datos:** contiene el código de la tecla pulsada.

**Registro de estado:** el bit 7 indica si el teclado está ocupado (valor 0) o si hay un dato válido en el registro de datos (valor 1).

**Registro de control:** posee las siguientes características:

- El bit 0 debe ponerse a 1 para iniciar una lectura. La escritura en este registro pone a 0 el bit 7 del registro de estado.
- El resto de los bits se utilizan para configurar ciertas características del teclado que no son necesarias para este ejercicio.

Suponga que los tres registros siempre se encuentran en direcciones consecutivas (tanto si son direcciones de E/S separada como si están mapeadas en memoria), en este orden: registro de datos, registro de estado y registro de control. Se pide:

**a)** Escriba una función en C que lea un dato del teclado y lo devuelva como resultado de la función. Los registros se encuentran **mapeados en memoria** en la dirección indicada por el parámetro de la función. Su prototipo es el siguiente:

*unsigned char lee_teclado( unsigned char *dir );*

**b)** Escriba una función en C que lea un dato del teclado y lo devuelva como resultado de la función. Los registros se encuentran en el **espacio de direcciones de E/S** en la dirección indicada por el parámetro de la función. Su prototipo es el siguiente:

*unsigned char lee_teclado( unsigned int puerto );*

**Ejercicio 25.** Se desea diseñar un sistema de control a distancia para un brazo biónico, de forma que nos permita manipular sustancias peligrosas sin necesidad de interacción alguna por parte del usuario. Se posee lo siguiente:
*Emisor: guante que recoge las posiciones de los dedos del usuario y la manda de manera inalámbrica. Se controla completamente por polling y utiliza registros de E/S aislada.*

**[0xF0]** Registro de Control (bits en blanco) y Estado (bits en gris):

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* |
|---|---|---|---|---|---|---|---|
| LEER | $DEDO_2$ | $DEDO_1$ | $DEDO_0$ | ON | ERROR | DATO | ON |

- Inicialización:
  - Si está no está activado el bit de "ON" de estado, activar el bit de "ON" de control.
  - Esperar hasta que se active el bit de "ON" de estado.
  - Comprobar que no hay error, en ese caso la rutina devuelve un '1', sino un '0'.
- Proceso de Lectura:
  - Colocar en los bits [$DEDO_2$, $DEDO_1$, $DEDO_0$] el número del dedo que se quiere leer (los dedos se identifican desde el "000" hasta el "100").
  - Activar el bit "LEER".
  - Esperar hasta que se active el bit de "DATO".
  - Leer del registro de datos la posición de dicho dedo.

**[0xF1]** Registro de Datos: devuelve la posición del dedo en grados (desde 0 hasta 180).

Para el emisor se pide:

**a)** Implemente la función de inicialización.

**b)** Implemente una función que reciba por parámetros un número entero (entre 0 y 4) que indique el dedo cuya posición se quiere leer, y que devuelva el valor leído.

**c)** Implemente la función principal del programa, en la que inicialice el dispositivo y lea secuencialmente las posiciones de los distintos dedos (uno a uno). Cada vez que recoja los cinco valores, haga una llamada a la función "*void enviaPosiciones(int d0, int d1, int d2, int d3, int d4)*", que manda de forma inalámbrica los valores leídos al receptor.

*Receptor: brazo biónico que recibe de forma inalámbrica las posiciones de los dedos del usuario y actúa sobre los motores de los dedos del brazo.*

**[0xE0]** Registro de Datos (escritura) del motor 0 (asociado al dedo 0 del emisor).
**[0xE1]** Registro de Datos (escritura) del motor 1 (asociado al dedo 1 del emisor).
**[0xE2]** Registro de Datos (escritura) del motor 2 (asociado al dedo 2 del emisor).
**[0xE3]** Registro de Datos (escritura) del motor 3 (asociado al dedo 3 del emisor).
**[0xE4]** Registro de Datos (escritura) del motor 4 (asociado al dedo 4 del emisor).

**[0xE5]** Registro de Datos (lectura): Almacena la información de la posición de un dedo. Esta información es recibida inalámbricamente por el emisor. Se reciben de forma secuencial desde el 0 hasta el 4 (cíclicamente); y es el programador el que tiene que controlar el valor de qué dedo está leyendo en cada momento.

**[0xE6]** Registro de Control (bits en blanco) y Estado (bits en gris):

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* |
|---|---|---|---|---|---|---|---|
| FIN_LECTURA | X | RESET | X | X | OTRO_DATO | ERROR | ON |

- Una lectura normal sería:
  - Esperar activación de "OTRO_DATO", que indica cuándo se ha recibido otro dato.
  - Realizar la lectura del registro de datos recibidos.
  - Activar el bit de "FIN_LECTURA".
  - Dependiendo del dedo cuya posición se haya leído, escribir en el motor correspondiente.
  - A continuación, se recibiría información del siguiente dedo (de forma cíclica): debe contabilizar qué dedo está leyendo en cada momento.

- Proceso de actuación tras "ERROR":
  - Resetear el dispositivo.
  - Comenzar a contabilizar desde el dedo 0.
- Control mediante E/S por interrupciones, siendo la señal de interrupción la producida por la activación del bit "OTRO_DATO" o por el bit "ERROR".

**d)** Para el receptor se pide implementar la rutina de tratamiento de la interrupción.

**Ejercicio 26.** Se pretende diseñar un sistema domótico. Para ello se dispone de un computador que se comunica con múltiples dispositivos de E/S mediante un protocolo propio. El sistema tiene la siguiente descripción:

**Registro Datos** *(accesible en la dirección REG_DATOS): entrada o salida dependiendo del comando utilizado.*

**Registro Control** *(accesible a través de la dirección REG_CONTROL):*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| $HAB_2$ | $HAB_1$ | $HAB_0$ | $TIPO_1$ | $TIPO_0$ | $ORDEN_1$ | $ORDEN_0$ | SEND |

- HAB2-0: indicación numérica de la habitación sobre la que actuar (desde la 0 hasta la 7).
- TIPO1-0: tipo de elemento sobre el que actuar. En nuestro sistema:
  - 0: luces.
  - 1: persianas.
  - 2: aire acondicionado.
  - 3: sensor de temperatura.
- ORDEN1-0: comando a indicar al elemento seleccionado. En nuestro sistema:
  - 0: apagar (en el caso de las persianas, cerrar completamente).
  - 1 encender (en el caso de las persianas, abrir completamente).
  - 2: configurar valor.
    - Aire acondicionado: indicar antes los grados en el registro de datos.
  - 3: leer valor.
    - Sensor de temperatura: ver temperatura de habitación (devuelto por el reg. de datos).
    - Resto: comprobar si encendidos (1) o apagados (0), devuelto por el reg. de datos.
- SEND: activar para mandar el comando.

**Registro Estado** *(accesible a través de la dirección REG_ESTADO):*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| X | X | X | X | X | X | ERROR | DATO |

- ERROR: comando erróneo.
- DATO: dato devuelto en el registro de datos. Se ha de esperar cada vez que se solicita un dato.

**a)** Como medida de seguridad, se quiere implementar la utilidad del apagado de luces de toda la vivienda. Implemente una función que reciba por parámetros el número de habitaciones que posee la vivienda y, en base a ellas, las recorra y apague las luces (únicamente en aquellas en las que se quedaron encendidas). La función devolverá el número de habitaciones que tenían las luces encendidas.

**b)** Realice un control por polling de la temperatura de todas las habitaciones (las 7), optimizando el consumo eléctrico. Para ello, inicialice los aires acondicionados de todas las habitaciones de su vivienda a una temperatura de 26 grados. Su programa consultará de forma continua la temperatura de los sensores de cada habitación; y en aquellas en las que se alcance la temperatura objetivo, se apagarán las máquinas de aire acondicionado. Si la temperatura sube de los 28 grados, se volverán a encender.

**Ejercicio 27.** Se quiere controlar un vehículo motorizado con un joystick analógico. Se dispone de un joystick conectado a un controlador de E/S, y un vehículo con dos motores conectados a otro (E/S separada en ambos casos). El problema se explicará y se resolverá por partes. Primero se controlará el joystick:

***PARTE 1: CONTROLADOR DEL JOYSTICK***

**Registro de datos** *(puerto 0x50 del mapa de E/S):*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EjeX2 | EjeX1 | EjeX0 | EjeY2 | EjeY1 | EjeY0 | Botón A | Botón B |

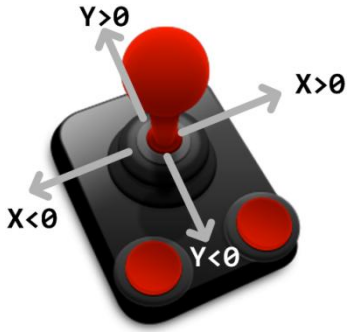
- EjeX2-0: valor que representa la posición X del joystick (valor con signo).
- EjeY2-0: valor que representa la posición Y del joystick (valor con signo).
- Botón A/B: pulsación del botón correspondiente.

**Registro de estado** *(bits sombreados) y control (puerto 0x51 del mapa de E/S):*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| X | X | X | On | Dato | Reset | Modo | Pedir |

- On: consultar si está encendido (1) o apagado (0).

- Dato: Si está a 1, hay datos nuevos disponibles en el registro de datos. En cuanto se lee el registro de datos, este bit se desactiva automáticamente.
- Reset: poner a 1 para arrancar o a 0 para apagar.
- Modo: poner a 0 para E/S por polling, a 1 para E/S por interrupciones.
- Pedir: poner a 1 para forzar una lectura de datos.




**a)** Implemente la función "**void leerValores()**". Esta función leerá los valores de *EjeX* y *EjeY* del registro de datos del joystick, siguiendo el protocolo de comunicación que se indica a continuación. Los valores obtenidos se deberán almacenar en las variables globales: **int ejeX** e **int ejeY**. ***Protocolo de comunicación:***
- Activación de la señal "Pedir" del registro de control.
- Espera a la activación de la señal "Dato" del registro de estado.
- Lectura de valores del registro de datos.

**b)** Implemente la función "**void inicializarJoystick()**", que iniciará el joystick de la siguiente forma: en primer lugar, debe cerciorarse de que está apagado. Posteriormente, configura el modo de E/S por polling, lo arranca y espera a que esté encendido.

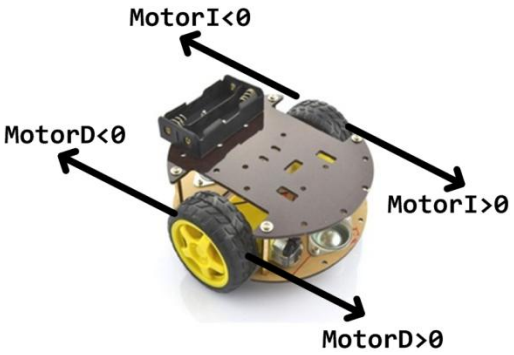
***PARTE 2: CONTROLADOR DE LOS MOTORES***

El control de los motores solo constará de un *registro de datos (puerto 0x52 del mapa de E/S)*:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| $MotorI_3$ | $MotorI_2$ | $MotorI_1$ | $MotorI_0$ | $MotorD_3$ | $MotorD_2$ | $MotorD_1$ | $MotorD_0$ |

- $MotorI_{3-0}$: potencia del motor izquierdo.
- $MotorD_{3-0}$: potencia del motor derecho.

Para mover el coche hacia delante, ambos motores deben tener el mismo valor. Si el motor derecho tiene más potencia, el coche girará a la izquierda, y viceversa.




El control, teniendo en cuenta los valores del joystick (X e Y), será:

- Valor del eje Y: potencia a dar A AMBOS MOTORES POR IGUAL. Nota: esto implica que si el eje X está centrado (es cero) irá en línea recta.
- Valor del eje X: especifica la variación de un motor respecto al otro. Casos:
  - Caso 1 – Y positivo y X positivo:
    - MotorI=Y+|X|
    - MotorD=Y

    *Ejemplo: ejeX=2, ejeY=3 → motorI = 3+|2| = 5, motorD = 3*
  - Caso 2 – Y positivo y X negativo:
    - MotorD=Y+|X|
    - MotorI=Y

    *Ejemplo: ejeX=-2, ejeY=3 → motorI = 3, motorD = 3 + |-2| = 5.*
  - Casos similares con Y negativo, según la lógica del movimiento.

*Nota: puede hacer uso de la función "abs(int valor)", que devuelve el valor absoluto de un entero.*

**c)** Implemente la función "**void potenciasMotores()**" que calcula la potencia a aplicar a cada motor según los valores de las variables globales *ejeX* y *ejeY* del joystick de la forma que se indicó antes. Tras ello, ambas potencias se escribirán en el puerto de E/S de control de los motores.

**d)** Implemente la función principal del programa, de forma que inicialice el joystick y, de manera continua, lea los valores del mismo y vaya modificando las potencias de los motores.

**Ejercicio 28.** El principal problema de los bisturís eléctricos es la posibilidad de generar una chispa en el interior del paciente debido a la concentración de gases, los cuales son generados por el propio bisturí en su acción de corte. Para evitar estos problemas, se realizan pausas periódicas durante las cuales se masajea la zona intervenida para vaciar al paciente de estos gases y poder continuar la operación.

Para evitar esto, se quiere implementar un sistema de aviso al cirujano ante el incremento de concentración de gases; y que, a la par, sea capaz de permitir la expulsión de estos de forma automática. Para ello, el nuevo

bisturí contendrá un sensor medidor de gases y un aviso lumínico al cirujano en caso de rebasar la concentración máxima segura de $CO_2$.

El microcontrolador empotrado en el bisturí que realizará dichas acciones posee los siguientes registros:

**[0xA0]** Registro de Control (bits en blanco) y Estado (bits en gris):

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* |
|---|---|---|---|---|---|---|---|
| READ | START-STOP* | RESET | ALARM | DATA | ERROR | BIST | ON |

- READ/DATA: READ solicita lectura de sensor, DATA indica que ya se puede leer de *Registro de Datos (1)*.
- RESET/ON: RESET reinicia el dispositivo (lo enciende si está apagado), ON indica el estado.
- ALARM enciende la señal lumínica (1) o la apaga (0).
- START-STOP*/BIST: START-STOP* enciende (1) o detiene (0) el bisturí; BIST indica su estado.
- ERROR indica error del dispositivo.
- Inicialización:
  - Si no está encendido, reiniciarlo y esperar a que se encienda.
  - Desactive la alarma y active el bisturí.
  - Si todo va bien, devuelva un '0' y, si ha habido error, devuelva un '1'.
- Proceso de Funcionamiento:
  - Activación de señal de lectura.
  - Espera a poder leer el valor del sensor.
  - Lectura del valor del sensor.
  - En base al valor leído, se actuará de la siguiente forma:
    - Si se rebasa el umbral perjudicial de $CO_2$ para la salud (5% o 50‰) se detendrá en seco el funcionamiento del bisturí y se activará la señal de alarma (en caso contrario deberían estar activos).
    - A partir del 0,5% (5‰) se activará el motor de extracción (solo necesario escribir un valor superior a 0 en Registro de Datos (2)): de forma que aumente su velocidad linealmente desde su valor mínimo (en el caso de sensor de gas al 5‰) a su valor máximo (en el caso de sensor de gas al 50‰).
    - Por debajo del 0,5% (5‰) el motor de extracción deberá estar parado y, por encima del 50‰, al máximo.

**[0xA1]** Registro de Datos (1): Valor del sensor de gas (en tanto por mil - ‰). Registro de lectura.

**[0xA2]** Registro de Datos (2): Velocidad del motor de extracción (en %). Registro de escritura.

**a)** Implemente la función de inicialización del dispositivo.

**b)** Implemente la función de tratamiento del dispositivo. Esta función realizará una lectura del sensor de gas, arrancará el motor de extracción a la velocidad calculada (según el valor leído) y activará las señales correspondientes en caso de sobrepasarse los límites. En este último caso, la función devolverá un '1' (0 en caso contrario).

En las intervenciones quirúrgicas, el equipo eléctrico de la mesa de operaciones está conectado a una toma de tierra situada en la base de la mesa (para mandar cualquier derivación al suelo de la sala de operaciones). Sin embargo, esta conexión suele ser un simple cable que, debido a su múltiple uso y delicadeza, pudiera deteriorarse o desconectarse. Para evitar estos problemas, el equipo posee un periférico que está continuamente controlando el consumo del equipo quirúrgico, así como diversas situaciones que pudieran ser problemáticas (como la pérdida de la toma de tierra).

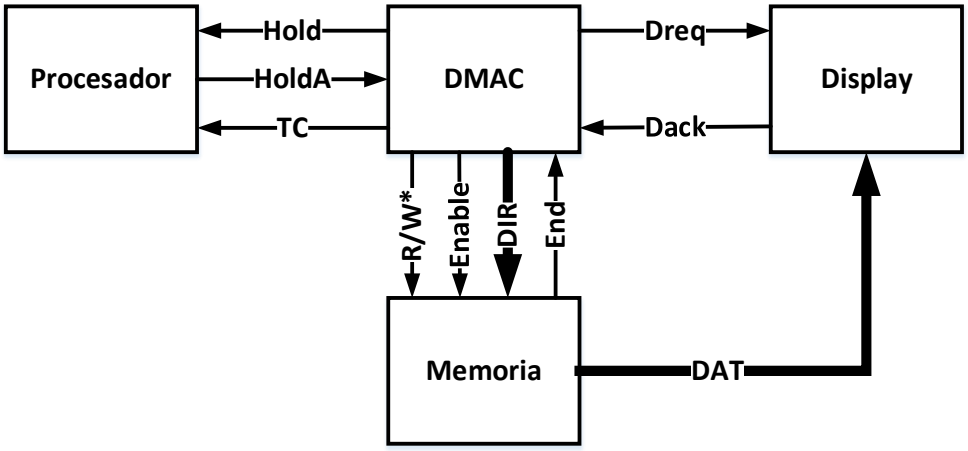
El **registro de Control y Estado** asociado a dicho periférico es el siguiente (dirección **0xA3**):

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* |
|---|---|---|---|---|---|---|---|
| INT_ERR | INT_GND | INT_DAT | INT_VCC | GND* | VCC | X | X |

- GND* y VCC indican (en caso de activas) que hay algún problema en la tierra o alimentación, respectivamente.
- Las señales INT_X son utilizadas para configurar X como fuente de interrupción del sistema. Así pues:
  - INT_ERR: entrada de interrupción el bit de ERROR del Registro 0xA0.
  - INT_GND: entrada de interrupción la señal GND* de este mismo registro.
  - INT_DAT: entrada de interrupción la señal DATA del Registro 0xA0.
  - INT_VCC: entrada de interrupción la señal VCC de este mismo registro.

**c)** Implemente la rutina de tratamiento de la interrupción asociada a todas las fuentes de interrupción salvo DATA. En caso de error del dispositivo deberá reiniciarlo. Si, por el contrario, la fuente de interrupción está relacionada con la tierra o alimentación del dispositivo, deberá parar el funcionamiento del bisturí y del motor; de igual forma, deberá activar la alarma lumínica.

**Ejercicio 29.** Se desea diseñar un DMAC empleando un microcontrolador. Este DMAC deberá comunicarse con un display externo, enviándole la información de los píxeles. El microcontrolador del DMAC dispone de un conjunto de registros para comunicarse con el display, con el procesador externo (para solicitar el acceso al bus) y con el controlador de memoria (para solicitar acceso a datos). El diagrama de bloques del sistema es el siguiente:




Nuestra tarea será programar el microcontrolador para que actúe como DMAC. Los registros que tiene son:

| Nombre registro | Puerto |
|---|---|
| Estado/Control Procesador | 0x26 |
| Estado/Control Memoria | 0x27 |
| Dirección Memoria | 0x28 |
| Control Display | 0x29 |
| Estado Display | 0x2A |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| X | X | X | HoldA | Hold | TC | X | X |
| X | End | R/W* | Enable | X | X | X | X |
| Dirección de acceso a memoria | | | | | | | |
| X | X | X | X | RST | X | X | Dreq |
| Dack | X | X | X | X | X | Err | X |

La funcionalidad de cada uno de los bits especificados en los registros de control/estado es la siguiente:

- Hold: solicita al procesador el acceso al bus (activo en alta).
- HoldA: el procesador concede acceso al bus (activo en alta).
- TC: aviso al procesador de la finalización de la transferencia del bloque completo (activo en alta).
- R/W*: especifica si la operación a realizar sobre la memoria es de lectura (1) o escritura (0).
- Enable: comienzo de la operación de lectura/escritura sobre la memoria (activo en alta).
- End: memoria avisa al DMAC del fin de la operación de lectura/escritura (activo en alta). En caso de lectura, el dato se encuentra disponible en el bus de datos.
- Dreq: avisa al display que hay nuevo dato disponible en el bus de datos para ser leído (activo en alta).
- Dack: el display notifica al DMAC que ya ha procesado el dato disponible en el bus (activo en alta).
- Err: el display indica una situación anómala en su funcionamiento (activo en alta).
- RST: reinicia el display (activo en alta). Utilizado en caso de error.

El funcionamiento del DMAC es el siguiente:

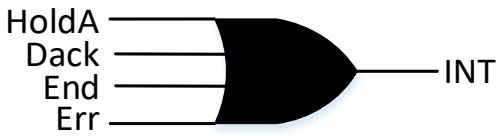
1. Solicita al procesador el acceso al bus (Hold).
2. Espera la concesión del bus (HoldA).
3. Coloca la dirección del dato a acceder en el registro de dirección de memoria.
4. Activa la señal de lectura de memoria (R/W*) y habilita el acceso a la misma (Enable).
5. Espera a la activación del fin de operación de memoria (End).
6. Avisa al display que hay un dato disponible en el bus (Dreq).
7. Espera a que el display termine de procesar el dato (Dack).
8. Desactiva la señal de dato disponible (Dreq).
9. Espera a la desactivación de Dack.
10. ¿Hay más datos para enviar?
    a. Sí → Volver al paso 2.
    b. No → Desactiva la solicitud de acceso al bus (Hold).
       Avisa al procesador que se ha completado la transferencia (TC).

**A tener en cuenta:**

- *Considere que la dirección base y el número de bytes a transferir se encuentran en las variables globales: dirbase y numBytes.*
- *Tenga en cuenta que los accesos a memoria del DMAC se realizan sobre direcciones consecutivas y que es tarea del DMAC generar las direcciones de memoria a las que se accede.*

**a)** Programe el código interno del DMAC para el control por polling del display. Para ello, implemente una función que lleve a cabo esta tarea, siguiendo los pasos indicados anteriormente.

**b)** Supongamos que el DMAC se encarga de controlar varios dispositivos externos; así pues, para eliminar las esperas activas producidas durante el control del Display, vamos a cambiar a un **control por interrupciones**. Suponga que el dispositivo se encuentra configurado y la interrupción registrada correctamente. Debe implementar únicamente la **rutina de tratamiento de la interrupción**, teniendo en cuenta que la interrupción hardware que recibe el DMAC es la siguiente:




**Ejercicio 30.** Se pretende implementar el sistema de protección de un procesador frente al aumento de la temperatura. Para ello, tendremos un periférico que se encarga de registrar las mediciones del sensor de temperatura interno al chip. El sistema de entrada se compone de un sensor de temperatura digital que es capaz de registrar y almacenar medidas (previa petición) en un registro de 8 bits (el sensor no registra temperaturas negativas).

Por otro lado, para proteger al procesador de este aumento de temperatura, tenemos acceso al registro que controla el divisor del reloj del sistema. De esta forma, si la temperatura sobrepasase ciertos límites, podríamos aumentar el valor de dicho registro e, inmediatamente, se reduciría la frecuencia de reloj del sistema. Obviamente, esto provoca una reducción en las prestaciones, pero disminuiría la disipación de calor del procesador.

**SENSOR DE TEMPERATURA**

**[0xA0] Registro de estado** (bits en cursiva) y **control** (bits en negrita):

| b7 | *b6* | *b5* | *b4* | *b3* | **b2** | **b1** | **b0** |
|---|---|---|---|---|---|---|---|
| X | *READY* | *NEW* | *ERROR* | *BUSY* | **REQ** | **RST** | **IRQ_EN** |

- IRQ_EN: permite configurar el método de E/S. 0: por polling; 1: con interrupciones.
- RST: permite dar un reset al sistema, si no está "READY".
- REQ: (=1) solicitud de dato al sensor de temperatura (control por polling).
- BUSY: (=1) indica si está ocupado. Antes de configurar o enviar una orden al dispositivo, hay que comprobar que no esté ocupado.
- ERROR: (=0) indica que todo va bien.
- NEW: (=0) indica que se ha producido una nueva lectura y se podrá leer en el reg. de datos.
- READY: (=1) indica que el dispositivo está funcionando.

**[0xA1] Registro de datos:** Dato de 8 bits (sin signo) que codifica la temperatura leída.

El proceso de lectura en polling es:

- Esperar si está "BUSY"
- Realizar una petición de dato.
- Esperar que nos avise de dato nuevo con "NEW".
- Leer dato del registro de datos.

**DIVISOR DE RELOJ DEL SISTEMA**

**[0xF0] Registro de estado** (bits en cursiva) y **control** (bits en negrita):

| b7 | *b6* | b5 | b4 | b3 | **b2** | **b1** | b0 |
|---|---|---|---|---|---|---|---|
| X | *PRIOR* | X | X | X | **MOD** | **UP/DOWN** | X |

- UP/DOWN: para indicar si aumentamos o disminuimos el divisor de reloj. Si queremos disminuir la frecuencia (para disminuir la temperatura) tendremos que aumentar el divisor (y viceversa). 0: aumentar; 1: disminuir.
- MOD: indicarle que modifique el divisor.

- PRIOR: (=1) hay una operación prioritaria ejecutándose y, mientras esté activa, no se puede modificar el divisor del reloj.
  El proceso de modificación en ***polling*** es:
  - Esperar a que no haya operación prioritaria ejecutándose ("PRIOR")
  - Indicar si va a ser una modificación de aumento o disminución del divisor ("UP/DOWN")
  - Activar bit "MOD"

**a)** Implementar la función de inicialización del sensor de temperatura. Deberá comprobar que está "READY" (si no, hará un "RESET" hasta que esté "READY") e indicarle el modo de funcionamiento. Luego, comprobará si ha habido algún error en la inicialización y devolverá ese resultado (0: no error, 1: error). Esta función recibe como parámetro de entrada un entero que indica si la E/S será por interrupciones (1) o por polling (0).

**b)** Implemente la función principal del sistema para controlarlo por polling. El sistema deberá:
- Iniciar el sensor de temperatura al inicio
- Repetidamente → Leer la temperatura
  - Si temperatura > 150º → aumentar el divisor de reloj.
  - Si temperatura < 100º → disminuir el divisor de reloj.

**Ejercicio 31.** Disponemos de una parcela en una zona rural. Allí hemos construido un pequeño huerto donde poder cultivar frutas y hortalizas. Sin embargo, como no acudimos con mucha asiduidad y la zona es bastante árida, nos interesa instalar un sistema de riego automático. Para ello, montamos tres sensores de humedad (en tres zonas diferentes del huerto) y un sistema de riego por goteo controlado a través de una bomba de agua. Para automatizar el sistema, todos los dispositivos los conectamos a un microcontrolador de bajo coste. Éste dispone de dos módulos de E/S conectados como E/S separada: uno para leer la información de los sensores de humedad y otro para controlar la bomba de riego. Resolveremos el ejercicio por partes:

***PARTE 1: CONTROLADOR DE LOS SENSORES DE HUMEDAD***

***Registro de estado*** *(puerto 0xF0 del mapa de E/S):*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| On0 | On1 | On2 | Err0 | Err1 | Err2 | Data Available | X |

- OnX: consultar si el sensor de humedad 'X' está encendido (1) o apagado (0).
- ErrX: consultar si el sensor de humedad 'X' ha dado un error (1) o no (0).
- Data Available: Indica si hay un dato nuevo (1) para leer en el registro de datos correspondiente.

***Registro de control*** *(puerto 0xF1 del mapa de E/S):*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reset0 | Reset1 | Reset2 | Read* | Sens[1] | Sens[0] | Mode | X |

- ResetX: arrancar/reiniciar (1) el sensor de humedad 'X', o apagarlo (0).
- Read*: (activo en baja) indicar que se quiere realizar una lectura (0).
- Sens[1-0]: colocar nº del sensor cuyo valor queremos leer (00: sensor1, 01: sensor2, 10: sensor3).
- Mode: indicar el modo de funcionamiento → 0: E/S por polling, 1: E/S por interrupciones.

***3 Registros de datos*** *(uno por sensor):*
- Sensor 0: puerto 0xF2 del mapa de E/S. Valor codificado como porcentaje.
- Sensor 1: puerto 0xF3 del mapa de E/S. Valor codificado como porcentaje.
- Sensor 2: puerto 0xF4 del mapa de E/S. Valor codificado como porcentaje.

***a)*** Implemente la función "int inicializarSensores(int modo)": inicializará el funcionamiento de los sensores. Para ello, debe seguir el siguiente protocolo para TODOS los sensores:
- Para cada sensor → Comprobar si el sensor está encendido en el registro de estado.
  - Si está apagado debe resetearlo y esperar a que esté encendido.
- Configurar el modo de funcionamiento (polling o int.) en base al parámetro recibido.
- Consultar si ha habido algún error en la inicialización en alguno de los sensores.
  - Solo si ha habido algún error, devolver un 1; en caso contrario devolver un 0.

***b)*** Implemente la función "int leerHumedad(int sensor)": leerá el valor del sensor que se pasa como parámetro (0, 1 o 2). El valor obtenido se devolverá al final. Comunicación:
- Colocar el número del sensor en los bits correspondientes del registro de control.
- Activación de la señal "Read*" del registro de control.
- Espera a la activación de la señal "Data Available" del registro de estado.
- Lectura del valor del registro de datos correspondiente y devolución del mismo.

### *PARTE 2: CONTROLADOR DE LA BOMBA DE AGUA*

***Registro de estado** (bits sombreados) y **control** (puerto 0xE0 del mapa de E/S):*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Online | Busy* | X | Pause | Play | X | X | X |

- Online: consultar si la bomba está arrancada (1) o pausada (0).
- Busy*: (activo en baja) consultar si el dispositivo está ocupado (0), en cuyo caso no puede configurarse.
- Pause: activar (1) para pausar el funcionamiento de la bomba; debe ponerse a 0 antes de activar Play.
- Play: activar (1) para arrancar la bomba; debe ponerse a 0 antes de activar Pause.

***Registro de datos** (puerto 0xE1 del mapa de E/S):* admite valores de potencia entre 0 y 255.

***c)*** Implemente la función "void bomba(int potencia)" que recibe como parámetro un porcentaje de potencia (0-100) para aplicar a la bomba de agua. El modo de activarla es el siguiente:
- Espera a que no esté ocupado (señal "Busy*").
- Desactiva la señal "Play" y Activa la señal "Pause".
- Espera a que "Online" esté desactivado.
- Coloca la potencia en el registro de datos: transformando del rango [0,100] al rango destino del registro de datos.
- Desactiva la señal "Pause" y Activa la señal "Play".
- Espera a que "Online" esté activado.

***d)*** Implemente la función principal, de forma que inicialice los sensores de humedad para funcionar por polling (si hay algún error, debe detener el funcionamiento). Debe leer continuamente los valores de humedad y activando la bomba de agua según lo siguiente:
- Si la media de la humedad es superior al 80% → Desactive la bomba (potencia al 0%).
- Si la media de la humedad es inferior al 70% →Activar la bomba a una potencia igual a diez veces la diferencia del porcentaje de humedad actual con el umbral (70%).

- Ejemplo:
  - Media de la humedad de los tres sensores: 65%
  - Diferencia con respecto al umbral: 70%-65% = 5%
  - Potencia del motor (10 veces la diferencia): 10*5% = 50%.

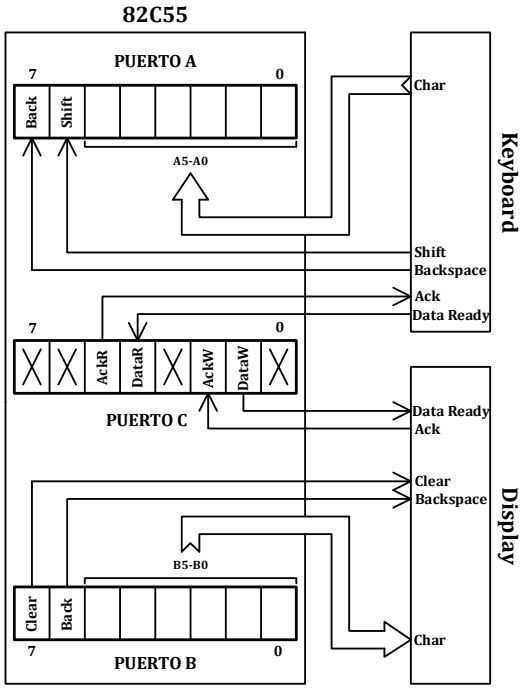
**Ejercicio 32.** Dispone de un controlador de E/S 82C55 de Intel que posee tres puertos: El puerto A y B son para enviar/recibir datos, y el puerto C actúa como registro de estado y control a la vez. A este controlador se le conecta un teclado (Keyboard) al puerto A, y una pantalla (Display) al puerto B (ver Figura).

La composición de los puertos es la siguiente:

**[dirección 0x10] Puerto A (lectura):**
- Bits A5-A0: contienen el carácter pulsado en el teclado.
- Bit A6: contiene información de la pulsación de la tecla "Shift".
- Bit A7: contiene información de la pulsación de "Backspace".

**[dirección 0x11] Puerto B (escritura):**
- Bits B5-B0: se almacena el carácter a pintar en el display.
- Bit B6: activar para borrar el último carácter del display.
- Bit B7: se puede activar para limpiar el display.

**[dirección 0x12] Puerto C (lectura [L]/escritura [E]):**
- Bit C0/C3/C6/C7: no utilizados.
- Bit C1: Comando al display para que pinte el carácter [E].
- Bit C2: Aviso del display para indicar que ya lo pintó [L].
- Bit C4: Aviso del teclado de que hay carácter disponible [L].
- Bit C5: Comando al teclado para indicar que ya se recogió [E].




Proceso de lectura del teclado:
- Keyboard coloca carácter en A5-A0
- Keyboard activa "Data Ready"
- CPU lee carácter.
- CPU activa "AckR".

Proceso de escritura en el display:

- CPU coloca carácter en B5-B0
- CPU activa "DataW"
- Display lee carácter.
- Display activa "AckW".

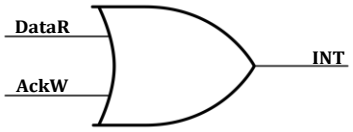
Comandos especiales:

- La activación del "Shift" cuando se realiza una lectura del teclado indica que el carácter debe pintarse en mayúsculas en el display (según la tabla ASCII, esto corresponde a escribir el mismo valor, pero restándole 32).
- La activación del "Backspace" en el teclado supondrá la activación de la misma señal en el display.
- La activación simultánea de "Shift" y "Backspace" supondrá la limpieza de pantalla en el display ("Clear").

Se pide:

**a)** Implemente el control por Polling completo del 82C55; de manera que esté leyendo continuamente los valores aportados por el teclado (así como los comandos especiales) y mostrando la información correspondiente por el display.

**b)** Teniendo en cuenta la entrada de interrupción indicada a continuación, implemente la rutina de tratamiento de interrupción del dispositivo.




**Ejercicio 33.** Dispone de un sistema empotrado que gobierna un vehículo aéreo no tripulado (UAV). Este vehículo puede ser controlado de forma manual o mediante vuelo autónomo. El sistema a implementar deberá ser capaz de actuar en base a los comandos de movimiento que vaya recibiendo por polling y, en caso de vuelo autónomo, gestionar la batería baja y enviar periódicamente los valores de los sensores de posición. El microcontrolador posee un controlador de entrada/salida separada con las siguientes características:

**Registro de Control (0x50):**

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| CMD1 | CMD0 | SEND | DATA3 | DATA2 | DATA1 | DATA0 | RETURN |

- CMD1-0: indicar el tipo de comando:
  - 0: roll, 1: pitch, 2: yaw, 3: power.
- DATA3-0: valor del comando.
- SEND: poner a '1' para enviar comando
- RETURN: poner a '1' para activar la vuelta a casa.

**Registro de Estado (0x51):**

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| C_READY * | BAT_LOW* | X | P_READY* | ERROR | X | X | X |

- C_READY*: vale '0' si ha terminado de procesar un comando y está disponible para recibir otro comando.
- BAT_LOW*: indicativo de batería baja ('0').
- P_READY*: vale '0' si hay datos posicionales para leer.
- ERROR: vale '1' si ha habido algún error con el comando recibido.

**Registros de Datos (0x52 – 0x53):**

- En estos registros se reciben las coordenadas GPS como Latitud (puerto 0x52) y Longitud (puerto 0x53).

**a)** Implemente una función para enviar un comando. La función recibe dos parámetros enteros: el primero indica el tipo de comando y el segundo el valor de dicho comando. Tenga en cuenta:

- Antes de enviar un comando, debe comprobar si el dispositivo está disponible para recibirlo.
- Para enviar un comando: indicar el tipo de comando (CMD1-0), el valor del comando (DATA4-0), y activar 'SEND'.
- La función debe devolver 1 si se ha producido un error con el comando, y un 0 si ha ido correctamente.

**b)** La información proveniente del mando de control se almacena en un vector de "char" de 4 posiciones (0: roll, 1: pitch, 2: yaw, 3: power). Así pues, implemente la función principal del programa para que, de forma continua, realice:
- Lea uno a uno los valores de dicho vector y los configure en el UAV mediante polling, usando la función realizada en el apartado anterior. Si se produce algún error, debe activar la vuelta a casa.
- Una vez termine de configurar los 4, si hay datos posicionales disponibles, debe leer el valor de Latitud y Longitud y almacenarlos en variables.

**c)** Implemente la rutina de tratamiento de interrupciones para este dispositivo para vuelo autónomo. Debe tener en cuenta que este dispositivo se supone ya configurado y posee tres fuentes de interrupción: batería baja (BAT_LOW*), datos posicionales para leer (P_READY*) y listo para recibir comando (C_READY*).
- En caso de datos posicionales disponibles, la rutina debe leer los valores de Longitud y Latitud y almacenarlos en variables globales.
- En caso de nivel bajo de batería, el sistema deberá configurar roll, pitch y yaw a '0' y power a '4' para reducir el consumo y mantener el vuelo. Se debe configurar primero roll y, en las diversas interrupciones producidas por C_READY, se configurará el resto de los valores. Este es el único caso en el que se puede activar la fuente de interrupción C_READY porque al ser vuelo autónomo, sólo se envían comandos en el caso de batería baja. Suponga que dispone de una variable global para gestionar el comando que se configura cada vez.

**Ejercicio 34.** Dispone de un microcontrolador y dos dispositivos, uno de ellos consta de un sensor de presencia y un sensor de luminosidad y el otro dispositivo es una luz de intensidad regulable. El sistema a implementar deberá gestionar el encendido de la luz cuando se detecte presencia con una intensidad inversamente proporcional a la luminosidad. El microcontrolador dispone de dos módulos de E/S conectados como E/S separada: uno para leer la información de los sensores de presencia y luminosidad y otro para controlar el encendido e intensidad de la luz regulable. Del apagado de la luz después de un cierto tiempo se encarga un temporizador externo al sistema a implementar.

*Características del controlador de E/S del sensor de presencia y sensor de luminosidad:*

**Registro de Control (0x50):**

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| RESET* | INT* | READ | X | X | X | X | X |

- RESET*: poner a '0' para activar/reiniciar el dispositivo.
- INT*: poner a '0' para activar modo por interrupciones, poner a '1' para activar modo por polling.
- READ: poner a '1' para solicitar lectura del sensor de luminosidad.

**Registro de Estado (0x51):**

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| X | X | X | ACTIVE | X | PRESENCE* | READY* | X |

- ACTIVE: vale '1' si el dispositivo está activo y '0' en caso contrario.
- PRESENCE*: vale '0' si el dispositivo detecta presencia.
- READY*: vale '0' si hay un dato que leer en el registro de datos.

**Registro de Datos (0x52):**
- En este registro se almacena el valor detectado por el sensor de luminosidad (valor entero positivo en el rango [0, 99]).

*Características del controlador de E/S de la luz regulable:*

**Registro de Control y estado (bit sombreado) (0x10):**

| *Bit 7* | *Bit 6* | *Bit 5* | *Bit 4* | *Bit 3* | *Bit 2* | *Bit 1* | *Bit 0* |
|---|---|---|---|---|---|---|---|
| IN3 | IN2 | IN1 | IN0 | SEND | X | X | ON |

- IN0-IN3: valor deseable de potencia de la luz regulable. Tiene 10 niveles.
- SEND: poner a '1' para encender la luz con la intensidad indicada en IN0-IN3.
- ON: vale '1' cuando se enciende la luz.

**a)** Implemente una función para la configuración inicial del sensor de luminosidad y presencia. Esta función recibirá un parámetro entero y deberá seguir el siguiente protocolo:

- Primero se debe comprobar si el dispositivo está inactivo, en tal caso se debe reiniciar y esperar a que esté activo.
- Configurar el modo de funcionamiento en base al parámetro: polling ('1') o interrupciones ('0'). En caso de habilitación por interrupciones, se deberá registrar con el vector de interrupción 0xF.

**b)** Implemente la función principal del sistema que configure su sistema y, de forma continua, compruebe la presencia y la luminosidad ambiental mediante polling. Cuando se detecte presencia, debe encender la luz con una intensidad inversamente proporcional a la luminosidad. Para ello siga el siguiente protocolo:

- Esperar mientras no se detecte presencia.
- Cuando se detecte presencia, leer el sensor de luminosidad y encender la luz mediante el siguiente proceso:
  - Solicitar la lectura del sensor de luminosidad.
  - Esperar la activación de 'READY*' que indica que hay un dato disponible para leer
  - Leer el valor del registro de datos.
  - Sabiendo que el valor de luminosidad es un valor entero positivo en el rango [0, 99] y la intensidad de la luz permite valores enteros positivos en el rango [1, 10], encienda la luz con un valor inversamente proporcional al valor de luminosidad.

*NOTA: para calcular el valor de potencia de la luz regulable a partir de la luminosidad ambiental deberá obtenerlo calculando el 10% de la luminosidad leída y aplicándole la inversa.*

**c)** Implemente la rutina de tratamiento de interrupciones para el mismo funcionamiento del apartado anterior. Debe tener en cuenta que el dispositivo se supone ya configurado y posee dos fuentes de interrupción: cuando se activa (en bajo) la señal PRESENCE* o la señal READY*.

- En caso de detección de presencia, la rutina debe leer solicitar la lectura del sensor de luminosidad.
- En caso de datos de luminosidad disponible, la rutina debe leer el valor del registro de datos y encender la luz con una intensidad inversamente proporcional al valor de luminosidad:

*NOTA: para calcular el valor de potencia de la luz regulable a partir de la luminosidad ambiental deberá obtenerlo calculando el 10% de la luminosidad leída y aplicándole la inversa.*