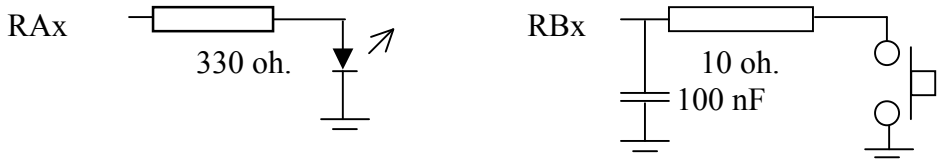
EJERCICIO 3 – INTERRUPCIONES-

CONEXIONES DE LA PLACA BASICA DE APENDIZAJE:

Para este ejercicio necesitamos la placa básica de aprendizaje, en ella tenemos conectado a las líneas RA0..RA3, 4 diodos led y a las líneas RB5, RB4 y RB0, tres pulsadores de la siguiente manera:



EJERCICIO 3-0:

Este ejercicio va a consistir en un contador cuyo valor se almacena en el puerto A, por lo que los leds nos indicarán el valor binario en cada momento. Para incrementar este contador utilizaremos una interrupción que se activará cada vez que pulsemos RB0.

Las interrupciones en el PIC16F84:

La ventaja de la utilización de interrupciones para detectar el estado de un pin, es que el programa no tiene que estar comprobando su estado continuamente; cuando el pin se activa se genera una interrupción, que interrumpirá el funcionamiento del programa, ejecutará el segmento de código apropiado para ese evento y volverá a donde estaba como si nada hubiera pasado.

Para gestionar las interrupciones, el PIC dispone del registro INTCON (0Bh, 8Bh) que posee la siguiente información:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

- **GIE:** Si lo ponemos a 1, activa el permiso para que puedan haber interrupciones. Si este bit esta a 0, aunque habilites otras interrupciones estas no serán permitidas.
- **EEIE:** Si lo ponemos a 1, permite que se genere una interrupción cuando haya termina de efectuar una escritura en la EEPROM.
- **T0IE:** Si lo ponemos a 1, permite una interrupción cada vez que el TMR0 pase de 255 a 0.
- **INTE:** Si lo ponemos a 1, permite la interrupción en el pin RB0/INT.
- **RBIE:** Si lo ponemos a 1, permite la interrupción por cambio en el estado de los pines RB7:RB4.
- **T0IF:** Si esta a 1, indica que se ha producido una interrupción en el TMR0.
- **INTF:** Si esta a 1, indica que se ha producido una interrupción en la patilla RB0/INT.

- **RBIF:** Si esta a 1, indica que se ha producido una interrupción porque al menos uno de los bits RB7:RB4 ha cambiado de estado.

Para, por ejemplo, gestionar la interrupción RB0/INT, tendremos que hacer lo siguiente:

1. Seleccionaremos el flanco de detección (si es necesario) mediante el bit INTEDG del registro OPTION_REG.
2. Activaremos la interrupción RB0/INT poniendo INTE a 1.
3. Activaremos las interrupción poniendo GIE a 1.

Cuando en el pin RB0/INT tenemos un cambio de señal cuyo flanco corresponde con el que hemos seleccionado en INTEDG, el microcontrolador almacenará en la pila la posición actual del PC y saltará a la posición de memoria 0004h, en ella tenemos que poner el segmento de código que utilizaremos para gestionar las interrupciones y que tienen que terminar con la instrucción RETFIE.

Si todas las interrupciones saltan a la posición 0004h, ¿Cómo podemos saber de quien es dicha interrupción?. Para esto en el registro INTCON, tenemos una serie de banderines, llamados flags (mira los bits del registro INTCON), que se pondrán a 1 para indicarnos de quien es la interrupción, cuando acabemos de gestionar esta interrupción y antes de ejecutar la instrucción RETFIE, tenemos que poner ese banderín a 0, si no, al salir se volverá a ejecutar la misma interrupción indefinidamente.

Para estudiar todo esto, utilizaremos el fichero Ejer3-0.asm cuyo código es el siguiente:

```
            Include p16f84.inc

            org H'0000
            goto Inicio             ;Se salta el vector de interrupción

            org H'0004              ;VECTOR DE INTERRUPCION
            btfsc INTCON,INTF       ;Comprueba si es int. de RB0/INT
            goto IntExt
            retfie
IntExt:     incf PORTA
            bcf INTCON,INTF
            retfie


Inicio:     Bsf STATUS,RP0          ;Activa la pagina 1
            Movlw b'11110000        ;RA0..RA4 como salida.
            Movwf TRISA
            Movlw b'11111111        ;Todo el puerto B como entrada.
            Movwf TRISB
            bcf OPTION_REG,NOT_RBPU ;Activa Pull-up en puerto B
            bcf OPTION_REG,INTEDG   ;Flanco de bajada en interrupcion.
            Bcf STATUS,RP0          ;Activa la pagina 0

            Movlw b'00000000        ;Leds apagados.
            Movwf PORTA             ;Transfiere W al puerto RA.

            bsf INTCON,GIE          ;Activa todas las interrupciones.
            bsf INTCON,INTE         ;Activa la interrupcion RA0/INT.

Fin:        Goto fin                ;Bucle de parada.
```

```
End
```

Pasemos a analizar el programa:

Como la rutina de gestión de interrupciones se sitúa en la posición 0004h, lo primero que tenemos que hacer es comenzar el programa con un salto para evitar escribir código en esa parte y luego poner una instrucción org H'0004 para situar el segmento de código correspondiente a las interrupciones justo en esa posición.

Bcf OPTION_REG,INTEDG: Con esta instrucción hacemos que la interrupción se active con el flanco de bajada, es decir, que la cuenta se incrementara al pulsar la tecla. Si lo ponemos a 1 el contador se incrementara al soltar la tecla.

Bsf INTCON,GIE: Activa todas las interrupciones.

Bsf INTCON,GIE: Activa la interrupción RB0/INT.

FIN: Goto FIN: Como la gestión de las interrupciones es independiente del funcionamiento del programa principal y en este no hacemos nada, dejamos al procesador en un bucle infinito.

Gestión de la interrupción:

Cuando se ejecuta la interrupción se realiza un salto a la posición 0004h, la rutina situada en dicha posición hace lo siguiente:

Btfsc INTCON,INTF: Esta instrucción testea el estado del bit INTF, este será 1 si la interrupción la ha generado RB0/INT, por lo que no se saltará la siguiente instrucción y hará un goto Intext. Si la interrupción la hubiera lanzado otro dispositivo se saltaría el goto y ejecutaría un RETFIE (observa que esta opción no es posible, salvo por un mal funcionamiento del chip ya que todas las demás interrupciones están inhibidas).

Bcf INTCON,INTF: Una vez ejecutada la rutina de control (incrementar PORTA), y antes de salir mediante un RETFIE, tenemos que poner el flag de INTF a 0 para que el microcontrolador sepa que esta interrupción ya ha sidon gestionada, si no lo hacemos, nada más salir el micro entenderá que la interrupción sigue vigente y la volverá a lanzar.

EJERCICIO 3-1:

Este ejercicio va a consistir en un contador cuyo valor se almacena en el puerto A, por lo que los leds nos indicarán el valor binario en cada momento. Para incrementar este contador utilizaremos una interrupción que se activará cada vez que cambie el estado de los pulsadores RB4 y RB5.

Para estudiar todo esto, utilizaremos el fichero Ejer3-1.asm cuyo código es el siguiente:

```
Include p16f84.inc

org H'0000
```

```
            goto Inicio

            org H'0004              ;VECTOR DE INTERRUPCION
            btfsc INTCON,RBIF       ;Comprueba int. cambio en B.
            goto IntExt
            retfie
IntExt:     incf PORTA
            bcf INTCON,RBIF
            retfie


Inicio:     Bsf STATUS,RP0          ;Activa la pagina 1
            Movlw b'11110000        ;RA0..RA4 como salida.
            Movwf TRISA
            Movlw b'11111111        ;Todo el puerto B como entrada.
            Movwf TRISB
            bcf OPTION_REG,NOT_RBPU ;Activa Pull-up del puerto B
            bcf OPTION_REG,INTEDG   ;Flanco de bajada en interrupcion
            Bcf STATUS,RP0          ;Activa la pagina 0

            Movlw b'00000000        ;Todos los led apagados.
            Movwf PORTA             ;Transfiere W al puerto RA.

            bsf INTCON,GIE          ;Activa todas las interrupciones.
            bsf INTCON,RBIE         ;Activa interrupcion cambio en B.

Fin:        Goto fin                ;Bucle de parada.

            End
```

Todo es igual que en el ejercicio anterior salvo que en vez de INTE e INTF usamos RBIE y RBIF. Cuando le des a los pulsadores los leds empezarán a contar muy deprisa hasta que sueltes, esto se debe a que el cambio permanece mientras estas pulsando el botón y la interrupción se ejecuta continuamente. El uso principal de esta interrupción es la de salir del modo standbye, por ejemplo si tenemos un mando a distancia con las teclas conectadas a los bits RB4..RB7, podemos dejar el circuito en standbye mediante la instrucción SLEEP, cuando pulsemos las teclas se activa el chip transmitiendo el código y luego vuelve al estado de standbye mediante SLEEP. Con esto podemos aumentar mucho la vida de la pila.