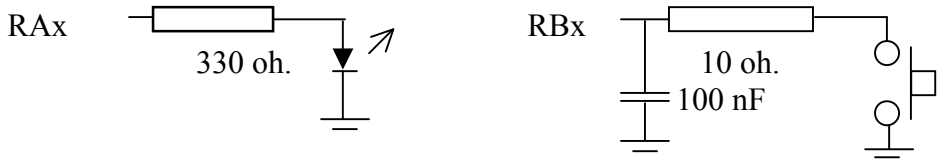
# EJERCICIO 1 – SALIDAS-

## 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 1-0:

Este ejercicio va a consistir en encender los led que nosotros decidamos. Para ello tendremos que indicarle al integrado que las líneas RA0..RA3 son de salida y cuales de ellas queremos que se pongan a 1 (led encendido) o a 0 ( led apagado).

Cada uno de los dos puertos RA y RB tienen dos registros asociados, PORTA y TRISA para RA y PORTB y TRISB para RB, los registros TRIS nos sirven para indicar si la línea es de entrada (poniendo un 1 en el bit del pin deseado) o salida (poniendo un 0 en el bit del pin deseado). Los registros PORT sirven para poner a 1 (poniendo un 1 en el bit del deseado) o a 0 (poniendo un 0 en el bit del pin deseado) el pin.

Para hacer esto hay que tener en cuenta una cuestión y es que los registros PORT y TRIS tienen la misma dirección pero en distintas paginas. Para poder pasar de una pagina a otra, tenemos que cambiar el estado del bit RP0 ( bit 5) de STATUS. Si lo ponemos a 1 estaremos en la pagina 1, si lo ponemos a 0 estaremos en la pagina 0.

ATENCIÓN: Si configuramos una línea como de salida y en ella conectamos un circuito que es también de salida, se producirá un cortocircuito que puede dañar el microcontrolador. Para evitar esto cuando se inicializa el micro, esté, se configura con todos sus pines de entrada TRISA=TRISB=FFh

Para nuestro ejercicio tenemos que irnos al registro de control del puerto A que es TRISA y programarle un 0 en las patillas que queramos que sean de salida, es decir, grabaremos B'11110000 y luego programaremos el valor que queramos que salga por los pines en el registro PORTA.

Esto lo haremos con las siguientes instrucciones:

```
;Ejercicio 1-0 de control de los puertos de salida

      Include p16f84.inc      ;Definición de registros.

      org H'0000
```

```
        Bsf STATUS,RP0          ;Activa la pagina 1
        Movlw b'11110000        ;Carga el registro W con la configuración
        Movwf TRISA             ;Transfiere ese dato a TRISA.
        Bcf STATUS,RP0          ;Activa la pagina 0

        Movlw b'00000011        ;Carga W con dos leds ON y dos OFF
        Movwf PORTA             ;Transfiere W al puerto RA.

Fin:    Goto fin                ;Bucle de parada.

        End
```

Pasemos a analizar el programa:

**Org H'0000**: Esta es una directiva del ensamblador que nos indica a partir de que posición queremos que se pongan las instrucciones que van a continuación. La posición 0000h es la posición de arranque del microcontrolador después del reset.

**Include p16f84.inc:** La directiva "include" del ensamblador inserta el fichero cuyo nombre viene como parámetro, en el ensamblado. El fichero p16f84.inc contiene la definición de todos los registros del PIC16F84 ( STATUS, PORTA, PORTB, TMR0, etc...) como constantes, de tal manera que cuando el ensamblador encuentra, por ejemplo, STATUS lo sustituye por el valor asignado (03h). Este fichero, que se encuentra en el directorio c:\PICmcb\Includes, lo puedes visualizar en el editor.

**Bsf STATUS,RP0:** Esta instrucción pone a 1 el bit RP0 del registro STATUS, activando la pagina 1 de la RAM.

**Movlw b'11110000 y Movwf TRISA:** Como el microcontrolador carece de una instrucción que asigne una constante a un registro f, para poder asignar a TRISA el valor de los pines que van de entrada y de salida, primero tenemos que asignar ese valor a W y luego asignar W a f.

**Bcf STATUS,RP0:** Volvemos a la pagina 0.

**Movlw b'00000011 y Movwf PORTA:** Cargamos en el puerto A el valor de los pines que queremos a 1 y cuales a 0.

**Fin: Goto fin:** La instrucción goto hace que el programa sigua en la dirección asignada por la etiqueta fin, que es ella misma, por lo que el programa ejecutará esta instrucción indefinidamente parándose la ejecución de este.

Este código lo tienes en Ejercicio1-0.asm y lo puedes ensamblar y ejecutar en la placa de aprendizaje.

Para ejecutar el programa en el ensamblador, vete a *Ficheros/proyecto* y pulsa el boton de Abrir/Nuevo. Busca el fichero Ejercicio1-0.asm que encontraras en el directorio C:\ProgMCB\Fuentes\Ejer01 y ábrelo, pulsa el botón de ensamblar y luego prográmalo en el chip ( asegúrate de que la configuración del chip sea 16F84 (XT) y que el puerto Lpt sea el adecuado) una vez programado puedes darle al botón de RUN para ejecutar el programa. Te aconsejo que pruebes a cambiar el valor de PORTA para ver el resultado.

ATENCIÓN: Cada vez que hagas una modificación, para que esta sea efectiva, tendrás que ensamblar y programar el chip de nuevo, antes de probarla.

**EJERCICIO 1-1:**

Este ejercicio va a consistir en hacer parpadear los 4 leds. Para ello, basándonos en los conocimientos adquiridos con el anterior ejercicio, utilizaremos el fichero Ejemplo1-1.asm cuyo código es el siguiente:

```
;Ejercicio 1-1 de control

      Include p16f84.inc      ;Definición de registros.

;CONSTANTES

      Tiempo1     equ H'02          ;Tiempo leds encendidos.
      Tiempo2     equ H'02          ;Tiempo leds apagados.

;VARIABLES

      RegRet0     equ H'0C
      RegRet1     equ H'0D
      RegRet2     equ H'0E

;PROGRAMA PRINCIPAL

            org H'0000

            Bsf STATUS,RP0          ;Activa la pagina 1
            Movlw b'11110000        ;Carga W con la configuración.
            Movwf TRISA             ;Cargalo en TRISA.
            Bcf STATUS,RP0          ;Activa la pagina 0

fin:        Movlw b'00001111        ;Carga W leds encendidos.
            Movwf PORTA             ;Transfiere W al puerto RA.
            movlw Tiempo1
            call Retardo            ;hace una espera
            Movlw b'00000000        ;Carga W con leds apagados.
            Movwf PORTA             ;Transfiere W al puerto RA.
            Movlw Tiempo2
            call retardo            ;hace una espera

            Goto fin                ;Repite el proceso.

; SUBRUTINAS

Retardo:    movwf RegRet2
Buret03:    movlw H'FF
            movwf RegRet1
Buret02:    movlw H'FF
            movwf RegRet0
Buret01:    decfsz RegRet0
            goto Buret01
            decfsz RegRet1
            goto Buret02
            decfsz RegRet2
            goto Buret03
            return

      End
```

Pasemos a analizar lo nuevo dentro de este fuente:

**Tiempo1 equ H'02:** La directiva EQU del ensamblador sirve para asignar a una etiqueta un valor, de tal forma que cada vez que el ensamblador se encuentre con la etiqueta la sustituirá por su valor. Por ejemplo, como Tiempo1 vale 02h, Movlw Tiempo1 asignará a w el valor 02h.

**RegRet0 equ H'0C:** Esta asignación mediante EQU, tiene la peculiaridad de que el valor asignado es una posición de memoria, con lo cual si escribimos incf RegRet0 haremos que el contenido de la posición 0Ch de la RAM se incremente en 1.

**Call Retardo:** Esta instrucción hace que el programa salte a la posición de memoria indicada en Retardo, almacenado previamente la posición desde donde ha realizado el salto, una vez ejecutada la subrutina se ejecuta una instrucción Return que hace que el programa continue desde donde salto ( es como una llamada a un procedimiento o a una función en un lenguaje de alto nivel).

**Retardo:** El código que compone la subrutina Retardo, lo que hace es un bucle de espera cuyo tiempo viene determinado por el valor de w al iniciarse la rutina. Como la velocidad del PIC es tan alta, he tenido que anidar tres bucles para que el tiempo sea apreciable. Para hacer un bucle con contador se procede de la siguiente forma:

1. Definimos un registro, en el que vamos a llevar la cuenta, por ejemplo RegRet0 y le asignamos una posición de memoria RegRet0 equ H'0C.
2. Le damos el valor Inicial al contador, por ejemplo, H'FF.
3. Ejecutamos la instrucción Decfsz RegRet0,f (como por defecto el ensamblador pone f como destino, podemos poner Decfsz RegRet0). Cada vez que ejecutemos esta instrucción decrementará RegRet0 y si no es 0 ejecutará la siguiente instrucción, que será un salto al bucle. Cuando sea 0 se la saltará finalizando el bucle.

Un bucle en ensamblador se ve así:

```
            movlw H'FF
            movwf RegRet0
Buret01:    ...instrucciones
            ...
            decfsz RegRet0
            goto Buret01
            ...
```

Con estos dos ejercicios hemos aprendido a utilizar los puertos como salidas, para poder controlar dispositivos externos.