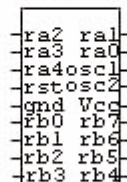


Practica con PICs

El primer paso importante es ver el diagrama de pines del PIC16F84, en el cual se observa como estan distribuidos sus pines. Este circuito integrado cuenta con 2 puertos configurables como entradas o salidas segun sea el caso y consta de 18 patas las cuales se encuentran asignadas de la siguiente manera:

1. Pata 1: -----RA2
2. Pata 2: -----RA3
3. Pata 3: -----RA4/TOCKI
4. Pata 4: -----Reset
5. Pata 5: -----Tierra (GND)
6. Pata 6: -----RB0/INT
7. Pata 7: -----RB1
8. Pata 8: -----RB2
9. Pata 9: -----RB3
10. Pata 10: -----RB4
11. Pata 11: -----RB5
12. Pata 12: -----RB6
13. Pata 13: -----RB7
14. Pata 14: -----Vcc
15. Pata 15: -----Osc2
16. Pata 16: -----Osc1
17. Pata 17: -----RA0
18. Pata 18: -----RA1

PIC16F84



El puerto A está denotado por el color Azul oscuro, el cual tiene sólo cinco pines que puedes configurar como entrada o salida. La pata 3, o sea, RA4/TOCKI puede ser configurado a su vez como entrada/salida o como temporizador/contador. Cuando es salida se comporta como colector abierto, por lo tanto debemos poner una resistencia Pull-up a Vcc de 1 Kohm. Cuando es configurada como entrada, funciona como disparador Schmitt Trigger por lo que puede reconocer señales con un poco de distorsión.

El puerto B está denotado por el color anaranjado, y tiene ocho pines que igualmente se pueden configurar como entrada o salida. Los pines 15 y 16 son únicamente para el oscilador externo el cual estudiaremos con más detalle más adelante. El pin 4, o sea, el Reset se debe conectar con una resistencia de 10 Kohm a Vcc para que el Pic funcione, si lo queremos resetear entonces pondremos un micropulsador con una resistencia de 100 Ohm a tierra.

La máxima capacidad de corriente para los puertos se muestra en la siguiente tabla:

	PUERTO A	PUERTO B
MODO SUMIDERO	80 mA	150 mA
MODO FUENTE	50 mA	100 mA

Por último tenemos los pines 14 y 5 que son la alimentación la cual no debe sobrepasar los 5 Voltios. Para esto nos aseguramos poniendo un regulador de voltaje (7805) en nuestro circuito.

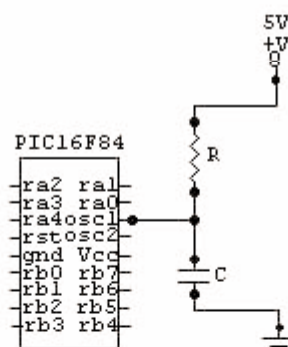
Es importante denotar que los pines de los puertos no utilizados los debemos conectar a +5V (Vcc) con una resistencia de 10 Kohm.

Oscilador Externo:

Es necesario para que nuestro PIC pueda funcionar, puede ser contactado de cuatro maneras diferentes. En la siguiente tabla encontraras los diagramas necesarios para su conexión y una breve descripción de cada uno.

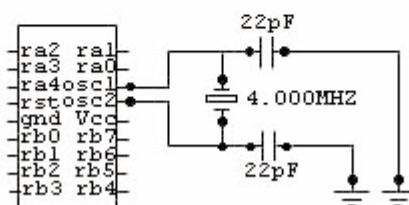
RC

Oscilador compuesto por una resistencia y un condensador.



XT

Oscilador compuesto por un cristal y dos condensadores.



LP

Oscilador compuesto por un cristal de baja frecuencia y bajo consumo de potencia.

El siguiente paso importante para tener claro como debemos empezar a programar es conocer la tabla de registros. Esta tabla está dividida en dos partes llamadas BANCO 0 y Banco 1. Nos debemos interesar momentáneamente en: STATUS, PORTA, PORTB, TRISA y TRISB.

Para que nuestro PIC pueda trabajar debemos configurar sus puertos según sea el caso, como entrada o como salida, haciendo antes la acotación que si le asignamos un CER0(0) a un pin éste será SALIDA y si asignamos un UNO (1) éste será ENTRADA.

Esta asignación de pines se hace programando los registros TRISA y TRIS B.

TRISA es el registro donde se almacenan los bits que asignan un pin como entrada o salida del PUERTO A. Recordemos que el puerto A sólo tiene 5 pines, por lo tanto un ejemplo de esto sería:

Si TRISA (puerto A) es igual a 00110 entonces esto se leería,

TRISA	ASIGNACION	ESTADO
RA0	0	SALIDA
RA1	1	ENTRADA
RA2	1	ENTRADA
RA3	0	SALIDA
RA4	0	SALIDA

El bit menos significativo se asigna desde RA0.

Si TRISB (puerto B) es igual a 00110010, entonces esto se leería,

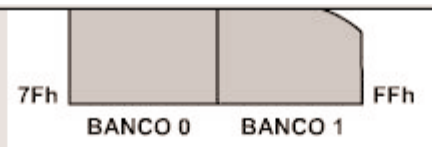
TRISB	ASIGNACION	ESTADO
RB0	0	SALIDA
RB1	1	ENTRADA
RB2	0	SALIDA
RB3	0	SALIDA
RB4	1	ENTRADA
RB5	1	ENTRADA
RB6	0	SALIDA
RB7	0	SALIDA

Ahora bien, pero como ponemos este número en TRISA y TRISB?

Para esto tenemos que ir a la tabla, la cual se divide en BANCO 0 y BANCO 1. Cuando el PIC arranca a correr el programa siempre se va a encontrar en el BANCO 0, por lo tanto debemos pasar al BANCO 1 para poder configurar los puertos asignando valores a TRISA y TRISB. Esto se logra a través del Registro STATUS, el cual nos servirá para cambiarnos de BANCO.

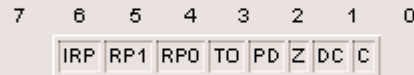
Memoria de Datos

00h	INDF	INDF	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h			87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2	89h
0Ah	PCLACTH	PCLATCH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	36 REGISTROS PROPOSITO GENERAL	MAPEADOS EN EL BANCO 0	8Ch
2Fh			AFh
30h			B0h



También es importante saber que este registro es de 8 BIT, o sea, ocho casillas, en la cual la No. 5 (RPO) define la posición del BANCO donde nos encontramos, por defecto siempre se encuentra en el BANCO 0.

Registro STATUS:



Si en la casilla 5 (RPO) del registro STATUS hay un CERO entonces estamos en el BANCO 0.

Si en la casilla 5 (RPO) del registro STATUS hay un UNO entonces estamos en el BANCO 1.

Pero como ponemos un UNO en la posición 5 del registro STATUS para entrar al BANCO 1?

Aqui es donde empezamos a ver las instrucciones de programa.

La dos primeras a utilizar son:

BSF que significa SET FILE REGISTER, es decir, pone un uno en la localización de la RAM especificada.

BCF que significa BIT CLEAR FILE REGISTER, es decir, pone un cero en la localización de memoria especificada.

Quiere decir entonces que para entrar al BANCO 1 tendríamos que poner un UNO en la posición 5 (RPO) del registro STATUS. La sintaxis sería:

bsf STATUS,5

Se lee, poner un UNO en la posición CINCO del registro STATUS. En este momento ya estamos dentro del BANCO 1.

NOTA: las intrucciones pueden se escritas en minusculas o mayusculas.

Ahora nos toca decidir según el proyecto que vallamos a hacer quien va a ser ENTRADA y quien va a ser SALIDA. Supongamos entonces que todos los pines del puerto A van a ser ENTRADA y el puerto B SALIDA.

Tentriamos que asignar al puerto A : **11111**

Y al puerto B : **00000000**

Movamos entonces estos valores a TRISA y TRISB respectivamente a través de la siguiente sintáxis:

movlw B'11111'

movwf TRISA

En la primera línea estamos moviendo 11111 a W. La W es el Registro de Trabajo, el cual usaremos para almacenar momentáneamente los datos que queramos mover. Después que los datos están en el registro de trabajo W, los podemos mover a TRISA, de esta manera ya configuramos el puerto A. La "B" y las comillas es la manera más común de designar el dato como NUMERO BINARIO, de esta manera se nos hace más fácil saber en determinado momento a quién pusimos como ENTRADA o SALIDA.

Ahora configuremos el puerto B.

movlw B'00000000'

movwf TRISB

Configurado el puerto B nos salimos del BANCO 1 al BANCO 0 para enpezar ya a programar.

Para salimos del BANCO 1 solo debemos poner un CERO en la posición 5 (RPO) del registro STATUS.

bcf STATUS,5

En este momento nos encontramos en el BANCO 0.

Práctica No.1

Empecemos entonces nuestro primer ejercicio!

El ejemplo más fácil y básico para aprender a manejar los puertos del PIC es encender y apagar uno o varios LED's.

Abre el editor de programa MPLAB y crea un nuevo proyecto.

Siempre que empecemos un programa debemos establecer como encabezado los registros con su dirección, dada al lado izquierdo de la tabla de registro. Es decir, pondremos primero los registros a utilizar con su respectiva dirección:

STATUS equ 0x3

PORTA equ 0x5

```
TRISA ..... equ ..... 0x5
PORTB ..... equ ..... 0x6
TRISB ..... equ ..... 0x6
```

; Los comentarios en un programa se inician con punto y coma (;).

; Declaración de Variables

```
RPO ..... equ ..... 5
```

```
RBO ..... equ ..... 0 ; Omita esta línea hasta que se le
..... ; indique en el transcurso del programa.
```

```
CONTADOR RES ..... 1 ..... ; Reserva un espacio de memoria a la
..... ; variable CONTADOR.
```

```
CONTADOR1 RES ..... 1 ..... ; Reserva un espacio de memoria a la
..... ; variable CONTADOR1.
```

; Estas dos líneas son variables definidas por el usuario reservadas para una subrutina de retardo que utilizaremos.

```
ORG ..... 0 ..... ; Hace referencia a una dirección
..... ; de memoria en el área de programa
..... ; a partir de la cual insertaremos
```

```
..... ; las instrucciones nemónicas que el
```

```
..... ; compilador deberá convertir.
```

; El siguiente paso es entrar al BANCO 1 para definir los puertos.

```
BSF ..... STATUS,RPO ..... ; Entramos al BANCO 1, RPO lo
..... ; declaramos arriba
..... ; y por lo tanto es igual a 5.
```

```
MOVLW ..... B'00000' ..... ; Movemos 00000 al registro de trabajo.
```

```
MOVWF ..... TRISA ..... ; Configuramos el puerto A como salida.
```

```
MOVLW ..... B'00000000' ..... ; Movemos 00000000 al registro de trabajo.
```

```
MOVWF ..... TRISB ..... ; Configuramos el puerto B como salida.
```

```
BCF ..... STATUS,RPO ..... ; Salimos del BANCO 1.
```

; Ahora podemos conectar a cualquiera de las salidas un LED, encenderlo y apagarlo. Detalles de conexión

; al final. Conectemos un LED a la salida RBO del puerto B, si lo quisieramos encender solo tendríamos que

; poner un UNO en la SALIDA RBO del puerto B. Como?..... asi:

```
INICIO ..... ; Etiqueta definida por el Usuario.
```

```
BSF ..... PORTB,0 ..... ; Usted podría poner también en
..... ; vez del cero la variable
```

```
..... ; declarada RBO, asegurandose que
```

```
..... ; esté en el encabezado.
```

```
..... ; Si lo desea agregue la línea que
```

```
..... ; omitió y cambie el cero
```

```
..... ; por RBO
```

; Cuando el PIC ejecuta la línea anterior pone un UNO en la salida,

; o sea 5 voltios, y por lo tanto el LED se enciende.

; Para hacerlo intermitente solo tendríamos que apagarlo y repetir

; la operación, pero antes de repetir debemos poner un retardo para

; poder ver el efecto intermitente.

; Esta rutina de retardo la insertaremos antes de apagarlo, luego lo

; apagamos e insertamos otra rutina de retardo.

; para esto utilizaremos una instrucción nueva: CALL, además debemos

; asignar un nombre propio a la subrutina,
 ; el que queramos, para poder decirle adonde debe ir. Cuando termine de
 ; ejecutar la subrutina de retardo se encontrará con la instrucción RETURN,
 ; devolviendocse así donde quedamos.

CALL RETARDO; Llama a RETARDO, va y lo ejecuta y
; luego se devuelve a la siguiente línea.

BCF PORTB,0; Apaga el LED.

CALL RETARDO; Llama a RETARDO

GOTO INICIO; Va a la etiqueta INICIO para mantener
; el LED intermitente,
; es decir, vuelve a empezar.

RETARDO

CALL RETARDO1

DECFSZ CONTADOR,1

GOTO RETARDO

MOVLW 80

MOVWF CONTADOR

RETURN

RETARDO1

DECFSZ CONTADOR1,1

GOTO RETARDO1

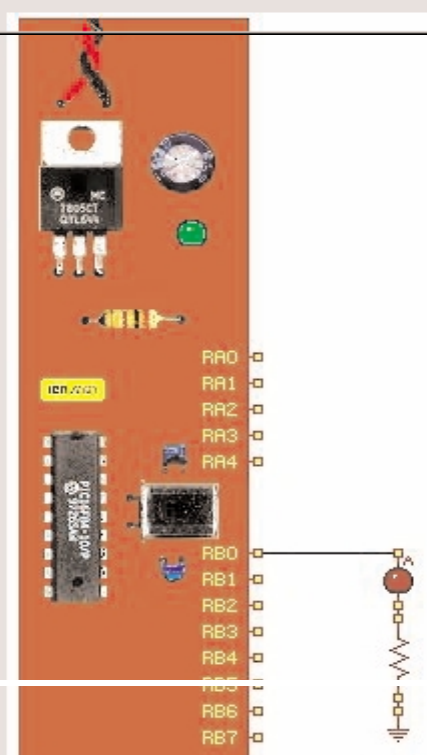
MOVLW 80

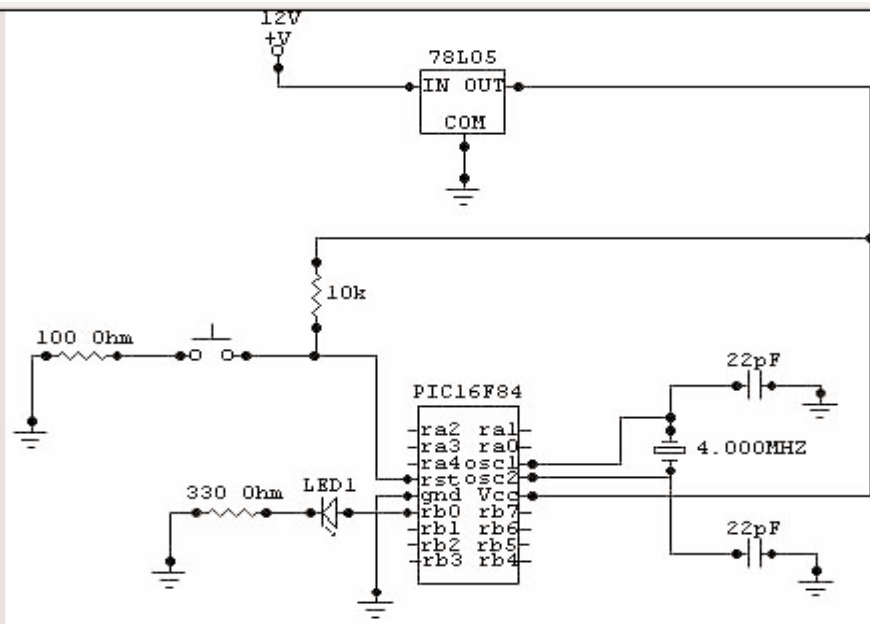
MOVWF CONTADOR1

RETURN

END

Asi se verá nuestro circuito:





Terminado el programa procedemos a compilarlo y luego a grabarlo en el PIC.