UNIDAD V

TEMPORIZADORES Y PUERTOS DE E/S

V.1. Temporizadores y contadores

Características generales

El free-run timer TMR0

Físicamente el timer es un registro cuyo valor se incrementa continuamente hasta 255 y a continuación vuelve a arrancar: 0, 1, 2, 3, 4...255...0, 1, 2, 3.....etcétera. .

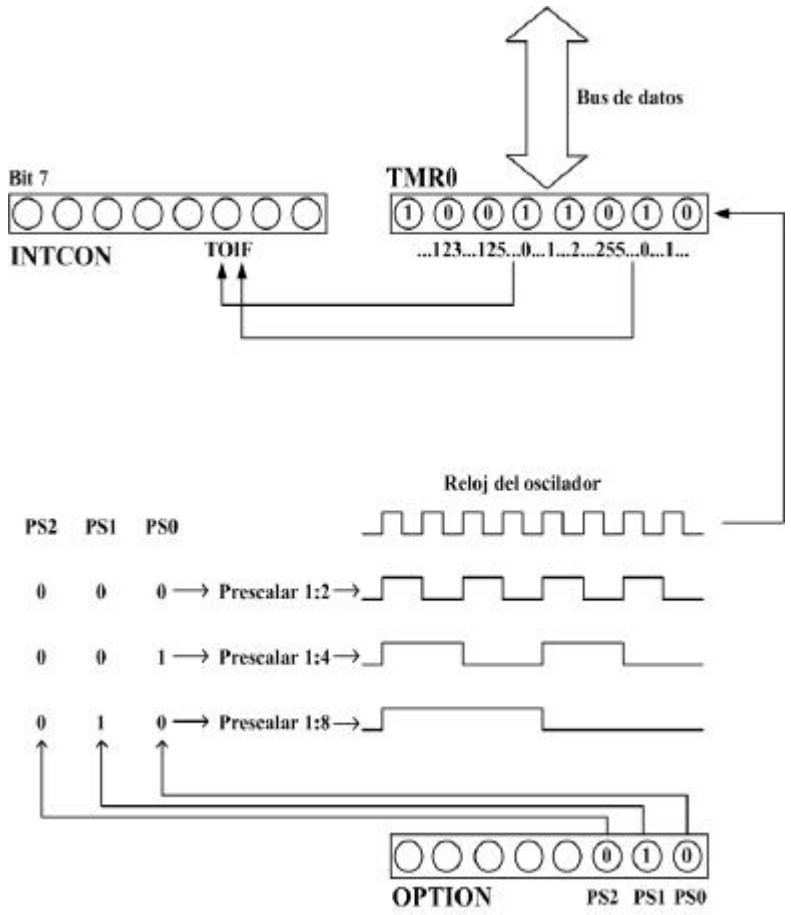


Figura V.1. Relación entre el timer TMR0 y el prescalar

Este incremento se hace en segundo plano (background) de todo lo que hace el microcontrolador. Una de las formas como el programador puede usar el timer es incrementando también una variable cada vez que ocurra un sobreflujo en el timer. Si se conoce cuanto tiempo necesita un timer para llegar de 0 a 255, entonces multiplicando el valor de una variable por ese tiempo se puede obtener el tiempo transcurrido.

Funcionamiento y modos de operación

El PIC16F84 tiene un timer de 8 bits. El número de bits determina hasta que valor contará el timer antes de iniciar el conteo desde cero nuevamente. En el caso de un timer de 8 bits, ese número es 255. En el diagrama anterior se muestra la relación existente entre un timer y un prescalar.

Prescalar es el nombre dado a la parte del microcontrolador que divide el reloj del oscilador antes de que pase a la lógica que incrementa el estado del timer. El número que divide al reloj se define por medio de los primeros tres bits del registro OPTION. El divisor más grande es 256. Esto significa que cada 256 pulsos de reloj se incrementa en uno el valor del timer. Esto es de gran ayuda en casos de necesitar medir periodos de tiempo grandes.

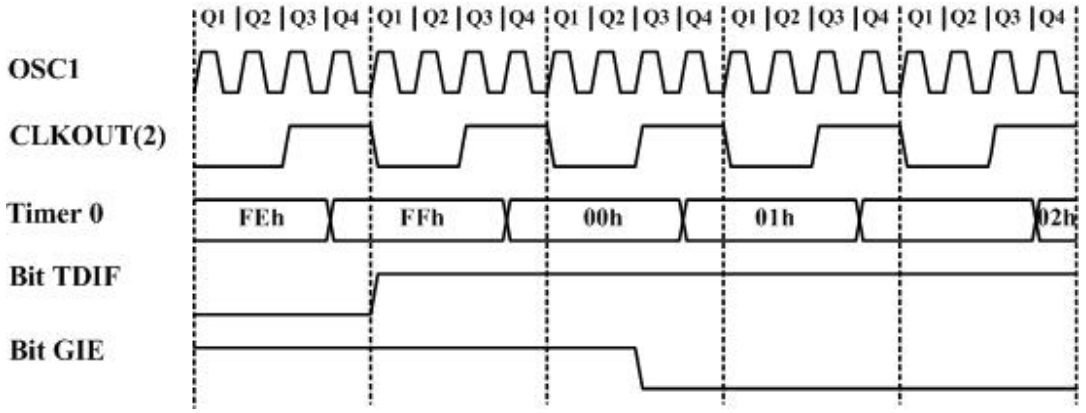


Figura V.2. Interrupción del timer TMR0

Después de cada conteo hasta 255 el timer reinicia su valor a cero y arranca un nuevo ciclo de conteo hasta 255. Durante cada transición de 255 a cero se activa el bit TOIF del registro INTCON. En al rutina de atención a la interrupción debe limpiarse el bit TOIF para que ocurra una nueva interrupción o que se pueda detectar un nuevo sobreflujo. Además del reloj del oscilador interno, se puede usar un reloj externo por medio de la línea RA4/TOCKI para incrementar el estado del timer. Elegir una de estas dos opciones se hace por medio del bit T0CS del registro OPTION. Además, la opción del reloj externo, se puede definir el tipo de transición (subida o bajada) por medio de la cual el timer incrementará su valor.

V.2. Programación de los temporizadores y contadores

El registro OPTION

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

R = Bit que se puede leer. W = Bit que se puede escribir.
U = Bit no usado, se lee un 0.
-n = Valor inicial al encender la fuente de alimentación.

Bits 0, 1 y 2-PS0, PS1 y PS2 (Prescaler Rate Select Bit-Bits de selección del prescalar)
Estos tres bits definen el valor del prescalar del Timer 0 (TMR0) y del watchdog.

Bits	TMR0	WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Bit 3-PSA (Prescaler Assignment Bit-Bit de asignación del prescalar)
Este bit asigna el prescalar al Timer 0-TMR0 o al watchdog.
1 = El prescalar es asignado al watchdog.
0 = El prescalar es asignado al free-run timer TMR0.

Bit 4-T0SE (TMR0 Source Edge Select Bit- Bit de selección del flanco o transición de la fuente de TMR0)

Se puede disparar TMR0 por medio de un pulso en la línea RA4/T0CKI, este bit determina si será con la transición de bajada o de subida.

1 = transición de bajada.

0 = transición de subida.

Bit 5-T0CS (TMR0 Clock Source Select Bit-Bit de selección del reloj de TMR0)

Este bit indica si el free-run timer incrementará su estado usando el oscilador interno cada 4 pulsos de la señal de reloj o por medio de pulsos externos en la línea RA4/T0CK1.

1 = Pulsos externos (contador).

0 = ¼ del reloj interno (temporizador).

Bit 6-INTEDG (Interrupt Edge Select Bit-Bit de selección del flanco o transición que dispara la interrupción)

Si la interrupción esta habilitada con este bit, es posible determinar el flanco o la transición con la que será activada la interrupción por medio de la línea RB0/INT.

1 = Transición de subida.

0 = Transición de bajada.

Bit 7-RBPU (PORTB Pull-up Enable Bit-Bit de habilitación de las resistencias de pull-up del puerto B)

Este bit activa o desactiva las resistencias pull-up internas del puerto B.

1 = Se desactivan las resistencias de pull-up.

0 = Se activan las resistencias de pull-up.

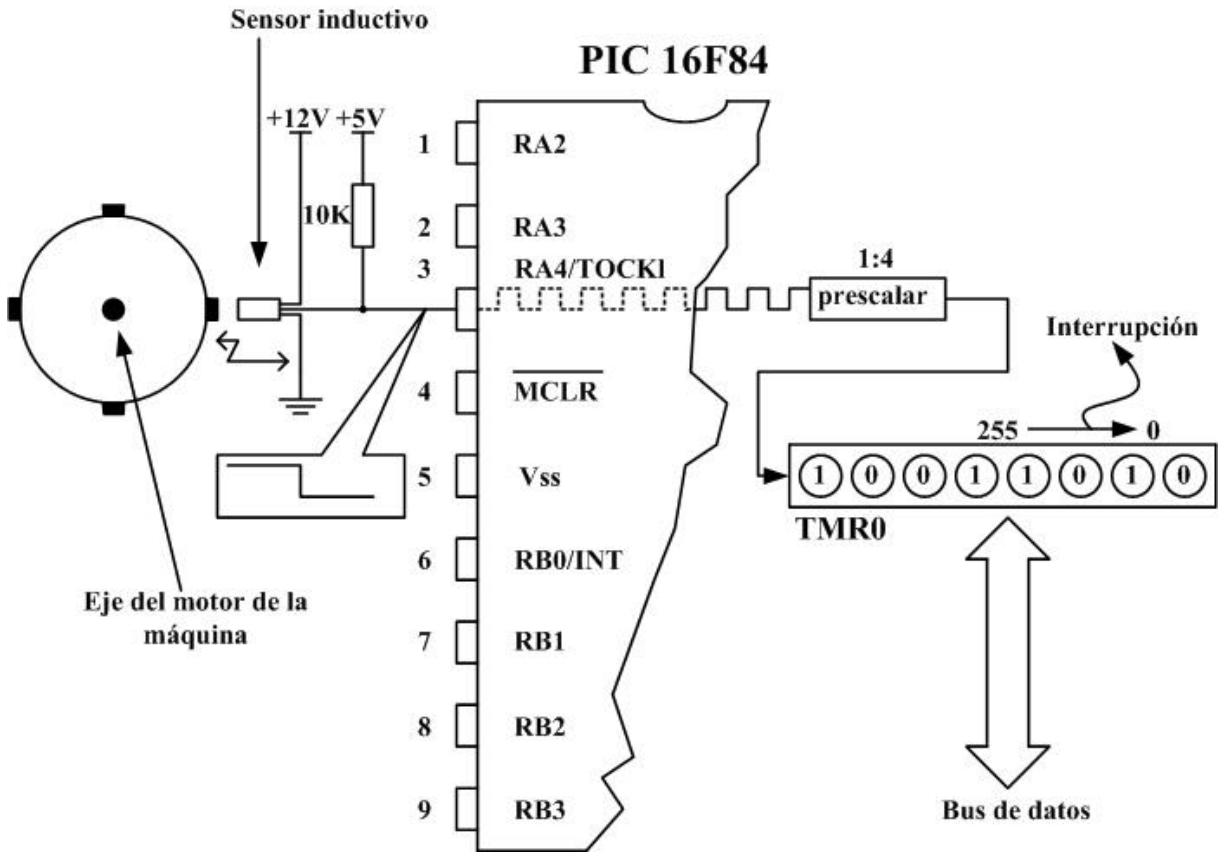


Figura V.3. Uso del timer para determinar el número de vueltas de un motor

En la figura anterior se muestra un ejemplo típico del uso del temporizador, donde se cuenta el número de vueltas completas que da un eje del motor de una máquina usando un reloj externo y el temporizador. Supóngase que se tiene un sensor inductivo a una distancia de 5 mm de la cabeza de los tornillos. El sensor inductivo generará una señal de bajada cada vez que la cabeza del tornillo esté paralela con la cabeza del sensor. Cada señal representará un cuarto de una vuelta completa y la suma de todas las vueltas completas se encontrará en el timer TMR0. El programa puede fácilmente leer este dato del temporizador por medio de un bus de datos.

A continuación se encuentra un programa que inicializa al temporizador para que se incremente con transiciones de bajada de la señal de reloj externa usando un prescalar de 1:4.

```
        clrf   TMR0             ;TMR0=0.
        clrf   INTCON           ;Interrupciones y TOIF=0
                                ;deshabilitadas.
        bsf    STATUS,RP0       ;Registro OPTION en banco 1.
        movlw B´00110001´       ;Prescalar 1:4, trans. de bajada,
                                ;reloj externo y pull up´s del
                                ;puerto B activados.
        movwf OPTION_REG        ;OPTION_REG <- W.

T0_OVL
        btfss INTCON,TOIF       ;Prueba bit de sobreflujo.
        goto T0_OVL             ;No ocurrió interrup., espera.
;
; (Parte del programa que procesa datos de num. de vueltas).
;
        goto T0_OVL             ;Espera nuevo sobreflujo.
```

Este mismo ejemplo se puede realizar usando una interrupción de la manera siguiente:

```
push    macro
        movwf   W_Temp          ;W_Temp <- W.
        swapf   W_Temp,F        ;Intercámbialos.
        BANK1                   ;Cambio al banco 1.
        swapf   OPTION_REG,W    ;W <- OPTION_REG.
        movwf   Option_Temp     ;Option_Temp <- W.
        BANK0                   ;Cambio al banco 0.
        swapf   STATUS,W        ;W <- STATUS.
        movwf   Stat_Temp       ;Stat_Temp <- W.
        endm                    ;Fin del macro push.
```

```
pop     macro
        swapf   Stat_Temp,W     ;W <- Stat_Temp.
        movwf   STATUS          ;STATUS <- W.
        BANK1                   ;Cambio al banco 1.
        swapf   Option_Temp,W   ;W <- Option_Temp.
        movwf   OPTION_REG      ;OPTION_REG <- W.
        BANK0                   ;Cambio al banco 0.
        swapf   W_Temp,W        ;W <- W_Temp.
        endm                    ;Fin del macro pop.
```

El prescalar puede ser asignado al timer TMR0 o al watchdog. El watchdog es un mecanismo que usa el microcontrolador para poder salir por si mismo de programas mal realizados o cuando ocurre una falla eléctrica. Cuando esto sucede, el microcontrolador detiene su trabajo y permanece en ese estado hasta que alguien o algo lo reinicialice (reset).

Una forma de reinicializarlo es usando el watchdog después de un cierto periodo de tiempo. El watchdog trabaja usando un principio muy simple: si ocurre un sobreflujo en el timer, el microcontrolador se reinicializa y empieza a ejecutar su programa nuevamente. De esta forma, un reset ocurrirá en caso de funcionamiento correcto o incorrecto del microcontrolador, por lo que el siguiente paso es escribir un uno en el registro WDT (con la instrucción CLRWDT) cada vez que ocurra un sobreflujo para evitar un reset en caso de un correcto funcionamiento. De esta forma, se evitará un reset mientras el programa se esté ejecutando correctamente, y en casos de mal funcionamiento se escribirá un cero en WDT y el PIC16F84 se reinicializará.

El prescalar es asignado al timer TMR0 o al watchdog timer por medio del bit PSA del registro OPTION. Limpiando el bit PSA el prescalar será asignado al timer TMR0. Cuando el prescalar es asignado al timer TMR0, todas las instrucciones que escriban al registro de TMR0 (por

ejemplo CLRF TMR0, MOVWF TMR0, BSF TMR0,...) limpiarán al prescalar también. Cuando el prescalar es asignado al watchdog timer, solo la instrucción CLRWDT limpiará al prescalar y al watchdog timer al mismo tiempo. El cambio del prescalar está completamente bajo control del programador y puede cambiarse en cualquier momento mientras un programa se esté ejecutando.

Resumiendo entonces, existe un solo prescalar y un timer. Dependiendo de las necesidades, se asignan ya sea al timer TMR0 o al watchdog.

V.3. La memoria de datos EEPROM

El PIC16F84 tiene una EEPROM de 64 bytes que comprende las localidades 00h a la 63h. La característica más importante de esta memoria es que no pierde su contenido cuando se apaga su fuente de alimentación. Los datos pueden ser retenidos por la EEPROM aun sin fuente de alimentación hasta por 40 años (según el fabricante del PIC16F84) y se pueden ejecutar hasta 10 000 ciclos de escritura.

La memoria EEPROM sirve, en la práctica. Para almacenar datos importantes o parámetros de un proceso, como por ejemplo la temperatura o nivel que serán controlados o ajustados por un regulador o un cierto proceso.

La EEPROM está colocada en un espacio especial de la memoria y se puede accesar por medio de los siguientes registros especiales:

EEDATA localizado en la dirección 08h, contiene los datos leídos o escritos.

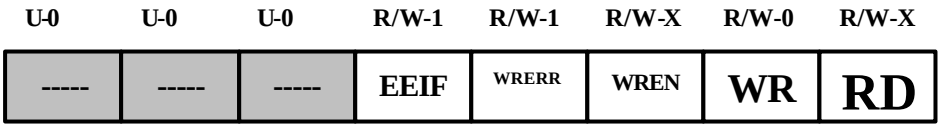
EEADR localizado en la dirección 09h, contiene la dirección de la localidad de EEPROM que será accesada.

EECON1 localizado en la dirección 88h, contiene bits de control.

EECON2 localizado en la dirección 89h. Este registro no existe físicamente y sirve para proteger a la EEPROM de escrituras accidentales.

El registro EECON1

Este registro es un registro de control localizado en la localidad de memoria 88h que contiene cinco bits funcionales. Los bits 5, 6 y 7 no se usan y leerlos dará como resultado cero.



R = Bit que se puede leer.	W = Bit que se puede escribir.
U = Bit no usado, se lee un 0.	
-n = Valor inicial al encender la fuente de alimentación.	

Bit 0-RD (Read Control Bit-Bit de control de lectura)
Activando este bit se inicia la transferencia de datos desde la dirección definida en el registro EEADR al registro EEDATA.
1 = Inicia la lectura.
0 = No inicia la lectura.

Bit 1-WR (Write Control Bit-Bit de control de escritura)
Activando este bit se inicia la escritura de datos desde el registro EEDATA hacia la dirección especificada por el registro EEADR.
1 = Inicia la escritura.
0 = No inicia la escritura.

Bit 2-WREN (EEPROM Write Enable Bit-Bit de habilitación de escritura a EEPROM)
Si este bit no está activado, el microcontrolador no permite la escritura a EEPROM.
1 = Escritura permitida.
0 = Escritura no permitida.

Bit 3-WRERR (Write EEPROM Error Flag-Bandera de error de escritura a EEPROM)
Este bit se activa solo cuando una escritura a EEPROM ha sido interrumpida por una señal de reset o cuando ha expirado el watchdog timer (si es que se encuentra activado).
1 = Ocurrió error.
0 = No ocurrió error.

Bit 4-EEIF (EEPROM Write Operation Interrupt Flag Bit-Bandera de interrupción de escritura de la EEPROM)
Este bit se usa para informar que ha terminado una escritura a la EEPROM. El programador debe limpiar este bit para permitir o detectar la terminación de otras escrituras a EEPROM.
1 = Terminó una escritura.
0 = No ha terminado aún una escritura o no ha iniciado.

Lectura de la memoria EEPROM

Activando el bit RD se inicia una transferencia de datos de la localidad cuya dirección se encuentra en el registro EEADR al registro EEDATA. En las lecturas de datos no se necesita tanto tiempo como en las escrituras, de tal forma que el dato leído y que se encuentra en el registro EEDATA se puede usar en la instrucción siguiente.

A continuación se muestra el segmento de un programa usado para leer datos de la memoria EEPROM.

```
bcf     STATUS,RP0    ;EEADR esta en la 09h, banco 0.
movlw   0x00          ;Direcc. de la localidad a leer.
movwf   EEADR         :Direcc. transferida a EEADR.
bsf     STATUS,RP0    ;EECON1 esta en la 88h, banco1.
bsf     EECON1,RD     ;Lectura de la EEPROM.
bcf     STATUS,RP0    ;EEDATA esta en la 08h, banco 0.
movf    EEDATA,W      ;W ✍ EEDATA.
```

Después de la ultima instrucción, el contenido de la dirección cero de la EEPROM se encuentra en el registro de trabajo W.

Escritura a la memoria EEPROM

Para escribir datos a una localidad de la memoria EEPROM, el programador debe escribir primero la dirección en el registro EEADR y el dato en el registro EEDATA. A continuación debe activarse el bit WR para llevar a cabo la escritura. El bit WR será limpiado y el bit EEIF se activará al terminar la escritura y podrá generar así una interrupción. Los valores 55h y AAh son las dos claves que deshabilitan la característica del PIC16F84 que evita que ocurran escrituras accidentales a la EEPROM. Estos dos valores deben escribirse al registro EECON2 que sirve únicamente para este propósito. Las líneas marcadas en el siguiente programa como 1, 2, 3 y 4 deben ejecutarse en ese orden en intervalos iguales de tiempo, por lo que es de suma importancia deshabilitar las interrupciones ya que éstas podrían cambiar los intervalos de tiempo necesarios para ejecutar estas instrucciones. Después de estas instrucciones y escribir a la EEPROM se pueden habilitar nuevamente las interrupciones.

Es recomendable que el bit WREN esté apagado todo el tiempo excepto cuando se escriban datos a la EEPROM, de tal forma que la posibilidad de escribir accidentalmente a esta memoria sea mínima. Todas las escrituras a la EEPROM limpiarán automáticamente la localidad antes de escribirla.

A continuación se muestra el segmento de un programa que escribe el dato 0xEE a la primera localidad de la EEPROM.

```
        bcf     STATUS,RP0      ;EEADR esta en la 09h, banco 0.
        movlw   0x00            ;Direcc. de la localidad a escribir.
        movwf   EEADR           :Direcc. transferida a EEADR.
        movlw   0xEE            ;Escribe el dato 0xEE .
        movwf   EEDATA          :El dato pasa a EEDATA.
        bsf     STATUS,RP0      ;EECON1 esta en la 88h, banco1.
        bcf     INTCON,GIE      ;Deshabilita todas interrupciones.
        bsf     EECON1,WREN ;Habilita escritura a EEPROM.
        movlw   0x55
1)      movwf   EECON2          ;Primera clave 0x55 -> EECON2.
2)      movlw   0xAA
3)      movwf   EECON2          ;Segunda clave
                                ;   0xAA-> EECON2.
4)      bsf     EECON1,WR       ;Inicia escritura.
        bsf     INTCON,GIE      ;Habilita interrupciones.
```

V.4. Puertos de entrada/salida

Características generales

Como un puerto nos referiremos a un grupo de líneas o pines del microcontrolador que pueden ser accesados simultáneamente, o en los que se puede establecer una combinación deseada de unos y ceros o leer por medio de ellos un cierto estado. Físicamente un puerto es un registro dentro del microcontrolador el cual está conectado a las líneas del mismo. Un puerto representa una conexión física de la CPU con el mundo externo. El microcontrolador usa los puertos para monitorear o controlar otros componentes o dispositivos. Algunas líneas tienen dos funciones, como por el ejemplo la línea PA4/TOCKI, el cual simultáneamente es la línea 5 del puerto A y puede operar como una línea de entrada para el contador (free-run counter). La selección de una de las dos funciones se realiza en uno de los registros de configuración de los puertos de entrada/salida.

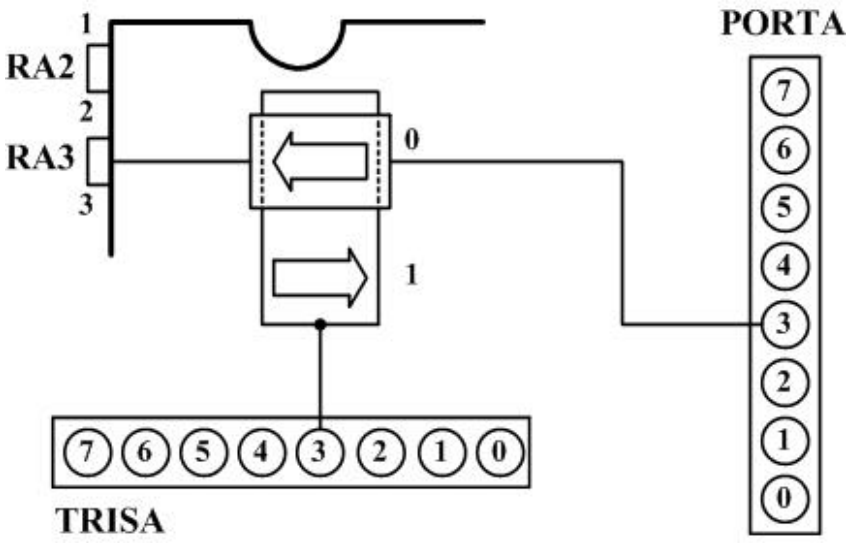


Figura V.4. Relación entre los registros TRISA y PORTA

Funcionamiento y modos de operación

Por medio de los registros TRIS del PIC16F84 se pueden definir las líneas de los puertos como entradas o como salidas, de acuerdo al dispositivo conectado a la línea correspondiente. Escribiendo un 1 lógico en el bit correspondiente del registro TRIS, la línea del puerto se configura como entrada, en caso contrario, un 0 lógico indicará que la línea es salida. Cada uno de los dos puertos del PIC16F84 tiene su correspondiente registro TRIS (TRISA y TRISB), localizados en las direcciones 85h y 86h, respectivamente.

V.5. Programación de los puertos

El puerto B

El puerto B del PIC16F84 tiene 8 líneas, su registro para configurar las líneas como entradas o salidas es el registro TRISB localizado en la dirección 86h. Cada línea del puerto B tiene una resistencia de pull-up interna (el resistor que define una línea a uno lógico), la cual puede activarse escribiendo un 0 lógico en séptimo bit (RBPU) del registro OPTION. Estas resistencias automáticamente se deshabilitan cuando la línea del puerto es configurada como salida. Cuando se alimenta el microcontrolador los pull-up´s se deshabilitan.

Las cuatro líneas más significativas del puerto B: RB4 a RB7, pueden generar una interrupción cuando su estado cambia de uno a cero lógico o de cero a uno lógico. Solo las líneas configuradas como entradas pueden generar la interrupción (si cualquiera de las líneas RB4-RB7 es configurada como salida, no se genera interrupción al cambiar su estado). Esta característica en conjunto con las resistencias de pull-up facilita la resolución de problemas prácticos como por ejemplo usar un teclado matricial. Si las filas del teclado se conectan a entradas del puerto B, al presionar una tecla se puede generar una interrupción. El

microcontrolador determina cual tecla se presionó al procesar la interrupción.

```
clrf    STATUS      ;Selección del banco 0.
clrf    PORTB       ;Puerto B=0.
bsf     STATUS,RP0  ;Selección del banco 1.
movlw   0x0F        ;Define las líneas de entrada y salida.
movwf   TRISB       ;Escribe al registro TRISB.
```

En el ejemplo anterior se muestra como las líneas 0, 1, 2 y 3 del puerto B se declaran como entradas y las líneas 4, 5, 6 y 7 como salidas.

El puerto A

El puerto A del PIC16F84 tiene cinco líneas, su registro para configurar las líneas como entradas o salidas es TRISA localizado en la dirección 85h. La quinta línea del puerto A tiene dos funciones. En esta línea está localizada también una entrada externa para el timer TMR0. Una de las dos funciones se selecciona con el bit T0CS (TMR0 Clock Source Selection bit) del registro OPTION. Este bit le indica al timer TMR0 que incremente su valor ya sea del oscilador interno o vía pulsos externos por medio de la línea RA4/T0CKI.

```
bcf     STATUS,RP0  ;Selección del banco 0.
clrf    PORTA       ;Puerto A=0.
bsf     STATUS,RP0  ;Selección del banco 1.
movlw   0x1F        ;Define las líneas de entrada y salida.
movwf   TRISA       ;Escribe al registro TRISA.
```

En el ejemplo anterior se muestra como las líneas 0, 1, 2, 3 y 4 del puerto A se declaran como entradas y las líneas 5, 6 y 7 como salidas.

Resumiendo, de lo que se ha visto hasta el momento se puede ya configurar el hardware para algunas líneas de los puertos del PIC16F84 funcionen como entradas y otras funcionen como salidas de la siguiente manera:

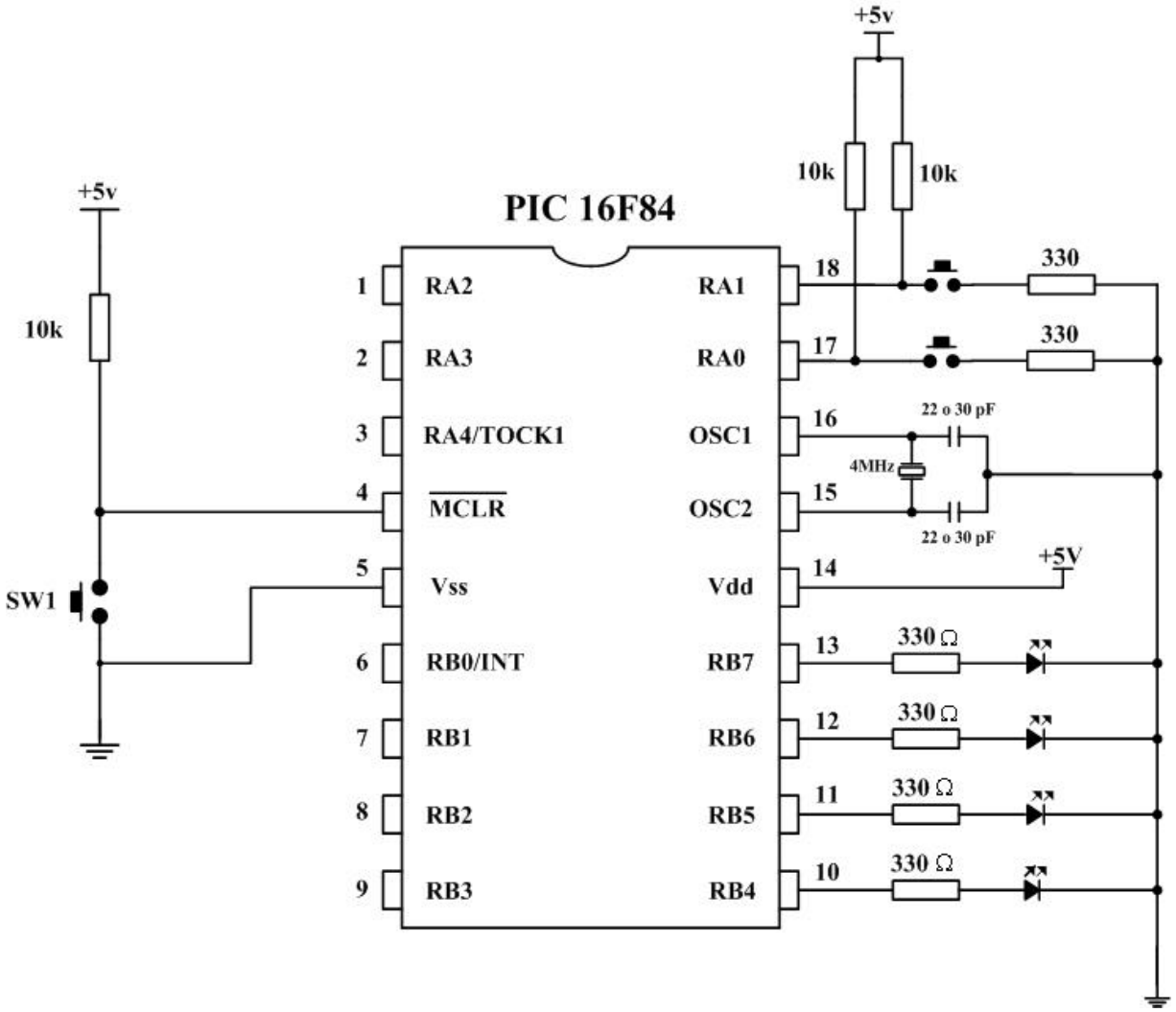


Figura V.5. Líneas de los puertos del PIC16F84 usadas como salidas y como entradas

A continuación se muestra el circuito de una fuente de alimentación para proporcionar una tensión estable de trabajo de +5 Vcc. a un sistema digital que use un PIC16F84.

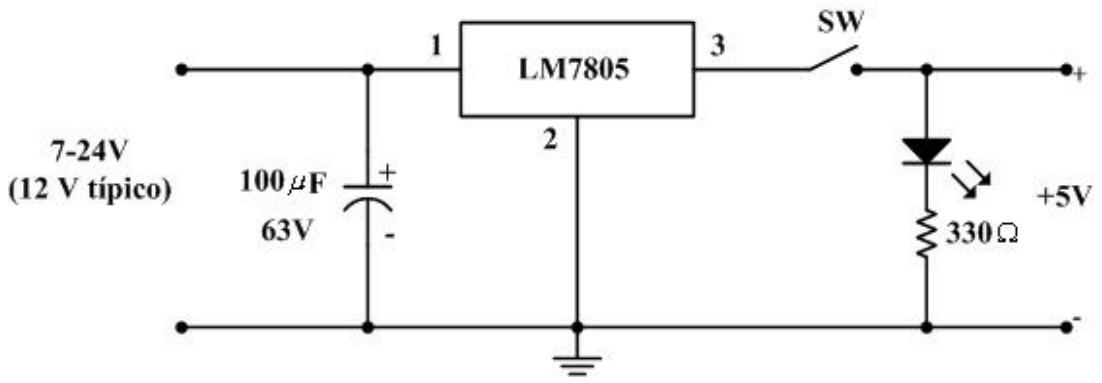


Figura V.6. Fuente de alimentación típica del PIC16F84