SISTEMAS ELECTRÓNICOS Y AUTOMÁTICOS
PRACTICAS DE MICROCONTROLADORES PIC

PRÁCTICA 5:
Medida del tiempo en un PIC

- ***TMR0***
- ***TMR1***
- ***Display 7 segmentos***

---

1. Objetivos:

- Gestión de los temporizadores en los PICs.

2. Temporizadores

Una de las labores más habituales en los programas de control de dispositivos suele ser determinar intervalos concretos de tiempo, y recibe el nombre de temporizador (TIMER) el elemento encargado de realizar esta función. También suele ser frecuente contar los impulsos que se producen en el exterior del sistema, y el elemento destinado a este fin se denomina contador.

Si las labores del temporizador o contador las asignamos al programa principal robarían mucho tiempo al procesador en detrimento de actividades más importantes. Por este motivo se diseñan recursos específicamente orientados a estas misiones.

2.1. El temporizador principal TMR0

El PIC 16F84 poseen un temporizador/contador de 8 bits, llamado TMR0, que actúa de dos maneras diferentes:

1 . Como ***contador*** de sucesos, que están representados por los impulsos que se aplican a la patita RA4/T0CKI. Al llegar al valor FFh se desborda el contador y, con el siguiente impulso, pasa a 00h, advirtiendo esta circunstancia activando un señalizador y/o provocando una interrupción.

2. Como ***temporizador***, cuando se carga en el registro que implementa al recurso un valor inicial se incremento con cada ciclo de instrucción (Fosc/4) hasta que se desborda, o sea, pasa de FFh a 00h y avisa poniendo a 1 un bit señalizador y/o provocando una interrupción. (Figura 31.)

Para que el TMR0 funcione como contador de impulsos aplicados a la patita T0CKI hay que poner a 1 el bit T0CS, que es el que ocupa la posición 5 del registro OPTION. En esta situación, el registro TMR0, que es el ubicado en la dirección 1 del banco 0 de la memoria de datos, se incrementa con cada flanco activo aplicado en la patita T0CKI. El tipo de flanco activo se elige programando el bit T0SE, que es el que ocupa la posición 4 del registro OPTION. Si T0SE = 1, el flanco activo es el descendente, y si T0SE=0, es el ascendente. Cuando se desea que TMR0 funcione como temporizador el bit T0CS=0.

En realidad, el PIC 16F84 y los de la gama baja disponen de dos temporizadores, el TMR0 y el Perro Guardián (Watchdog). El primero actúa como principal y sobre él recae el control de tiempos y el contaje de impulsos. El otro vigila que el programa no se «cuelgue», y para ello cada cierto tiempo comprueba si el programa se está ejecutando normalmente. En caso contrario, si el control está detenido en un bucle infinito a la espera de algún acontecimiento que no se produce, el Perro Guardián «ladra», lo que se traduce en un Reset que reinicializa todo el sistema.

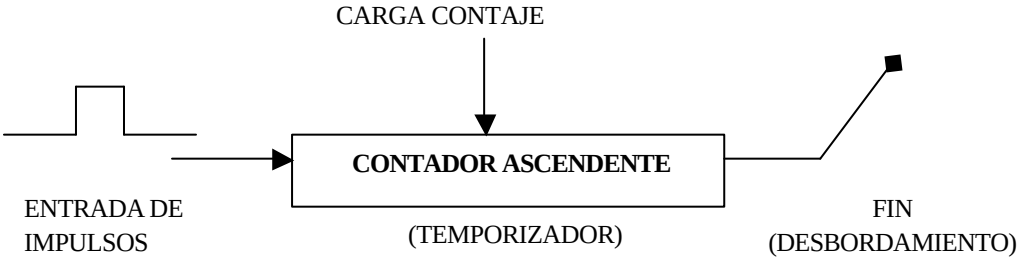


Figura 1. *Esquema simplificado de un temporizador/contador. Como temporizador se carga con un valor inicial que se va incrementando hasta el desbordamiento. Como contador va incrementando su valor con cada impulso que se le aplica. Al alcanzar el máximo valor binario se «desborda» y pasa a cero, circunstancia que indica un señalizador.*

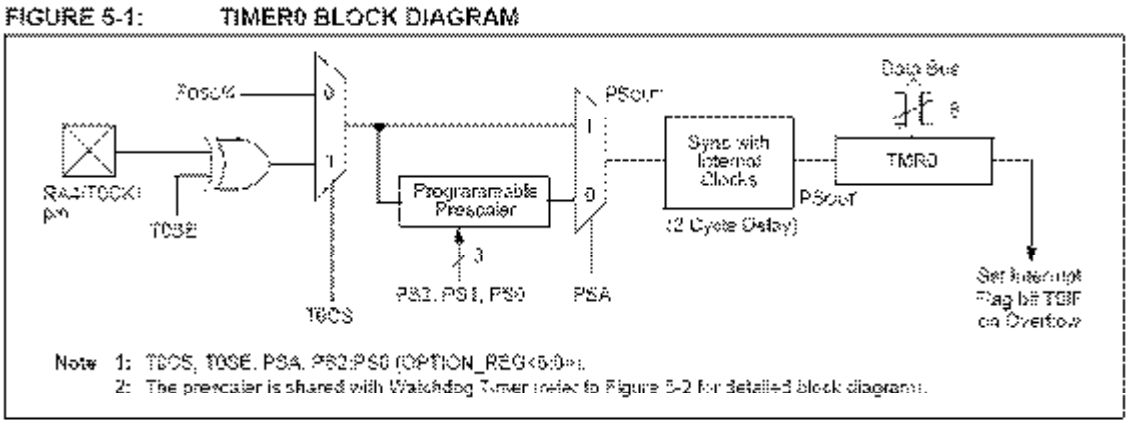


Figura 2. *Diagrama de bloques del sistema TMR0.*

A menudo el TMR0 y el Perro Guardián precisan controlar largos intervalos de tiempo y necesitan aumentar la duración de los impulsos de reloj que les incrementa. Para cubrir este requisito se dispone de un circuito programable denominado Divisor de frecuencia, que divide la frecuencia utilizada por diversos rangos.

Para programar el comportamiento del TMR0, el Perro Guardián (WDT) y el Divisor de frecuencia se utilizan algunos bits del registro OPTION y de la Palabra de Configuración.

El Divisor de frecuencia puede usarse con el TMR0 o con el WDT. Con el TMR0 actúa como Predivisor, es decir, los impulsos pasan primero por el Divisor y luego se aplican al TMR0, una vez aumentada su duración. Con el Perro Guardián actúa después, realizando la función de Postdivisor. Los

impulsos, que divide por un rango el Divisor de frecuencia, pueden provenir de la señal¡ de reloj interna (Fosc/4) o de los que se aplican a la patita T0CKI.

El TMR0 se comporta como un registro de propósito especial (SFR) ubicado en la dirección 1 del banco 0 de la memoria de datos. En igual dirección, pero en el banco 1, se halla el registro OPTION.

TMR0 puede ser leído y escrito en cualquier momento al estar conectado al bus de datos. Funciona como un contador ascendente de 8 bits. Cuando funciona como temporizador conviene cargarle con el valor de los impulsos que se quiere temporizar, pero expresados en complemento a 2. De esta manera, al llegar el número de impulsos deseado se desborda y al pasar por 00 H se activa el señalizador TOIF y/o se produce una interrupción.

Para calcular los tiempos a controlar con TMR0 se utiliza la siguiente fórmula práctica:

Temporización=T_{CM} · Prescaler · (256-Carga TMR0)

donde

- ***Temporización*** es el tiempo deseado.
- ***T_{CM}*** es el período de un ciclo máquina. Para 4 MHZ, TCM=1µs.
- ***Prescaler*** es el rango de divisor de frecuencia elegido.
- ***(256-Carga TMR0)*** es el número total de impulsos a contar por el TMR0 antes de desbordarse en la cuenta ascendente.

En cualquier momento se puede leer el valor que contiene TMR0, sin detener su contaje. La instrucción adecuada al caso es *movf TMR0,W.*

El registro OPTION

La misión principal de este registro es controlar TMR0 y el Divisor de frecuencia. Ocupa la posición 81h de la memoria de datos, que equivale a la dirección 1 del banco 1. El bit T0CS (Timer 0 Clock Edge Select) selecciona en el multiplexor MPX1 la procedencia de los impulsos de reloj, que pueden ser los del oscilador interno (Fosc/4) o los que se aplican desde el exterior por la patita T0CKI. El bit T0SE (Timer 0 Clock Source Select) elige el tipo de flanco activo en los impulsos externos. Si T0SE = 1 el flanco activo es el descendente y si T0SE = 0 el ascendente.

El bit PSA del registro OPTION asigna el Divisor de frecuencia al TMR0 (PSA = 0) o al WDT (PSA = l).

Los 3 bits de menos peso de OPTION seleccionan el rango por el que divide el Divisor de frecuencia los impulsos que se le aplican en su entrada.

El bit 6 INTEDG *(Interrupt Edge)* sirve para determinar el flanco activo que provocará una interrupción externa al aplicarse a la patita RB0/INT. Un 1 si es ascendente y un 0 descendente.

El bit 7 RBU (*RB Pull-Up*) activa, si vale 0, o desactiva, si vale 1, las resistencias Pull-Up que pueden conectarse a las líneas de la puerta B.

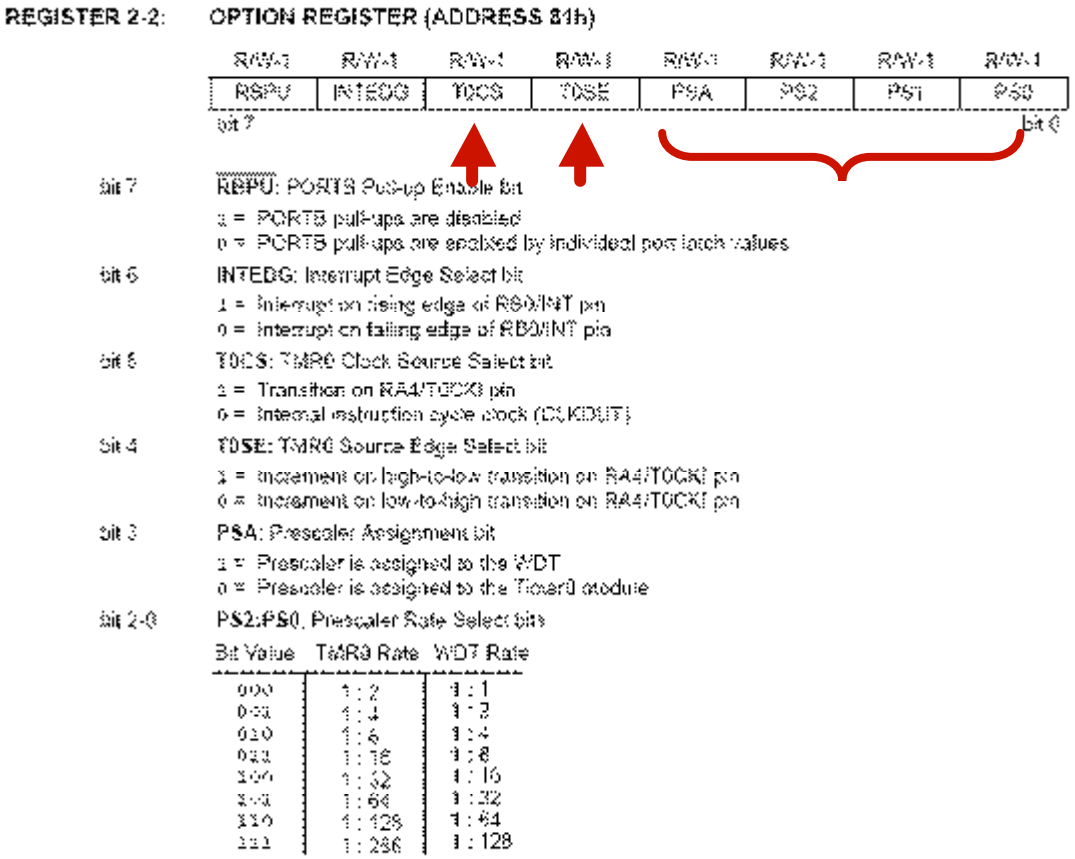


Figura 3. *Bits del registro OPTION en el PIC16F84.*

El registro INTCON

El flag de fin de contaje T0IF se encuentra en el registro INTCON.

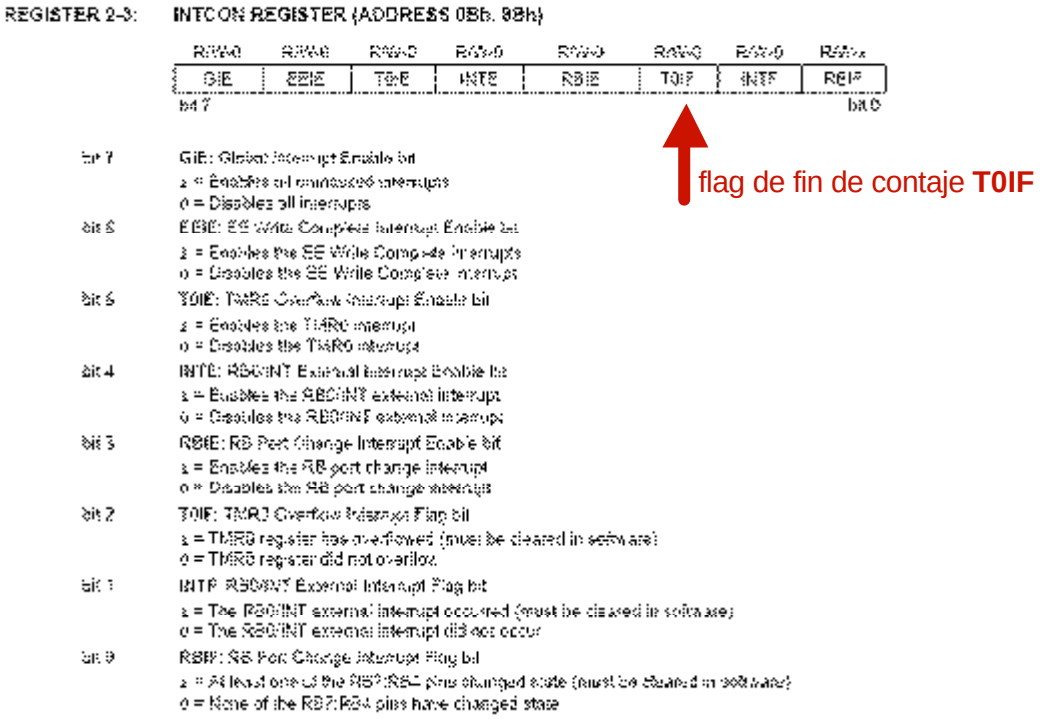


Figura 4. *Bits del registro INTCON en el PIC16F84.*

2.2. El perro guardián

Se trata de un contador interno de 8 bits que origina un Reset cuando se desborda. Su control de tiempos es independiente del TMR0 y está basado en una simple red RC. Su actuación es opcional y puede bloquearse para que no funcione programando el bit WDTE de la Palabra de Configuración.

Para evitar que se desborde el Perro Guardián hay que refrescarle previamente. En realidad este refresco consiste en ponerle a cero mediante las instrucciones *clrwdt* y *sleep*. El programador debe analizar las instrucciones de la tarea y situar alguna de esas dos en sitios estratégicos por los que pase el flujo de control antes que transcurra el tiempo asignado al WDT. De esta manera si el programa se «cuelga» no se refresca el Perro Guardián y se produce la reinicialización del sistema.

La instrucción clrwdt borra al WDT y reinicia su cuenta. Sin embargo, la instrucción sleep, además de borrar WDT detiene al sistema y lo mete en un estado de «reposo» o «de bajo consumo». Si no se desactiva el Perro Guardián al entrar en el modo de reposo, al completar su contaje provocará un Reset y sacará al microcontrolador del modo de bajo consumo. Para desactivar al Perro Guardián hay que escribir un 0 en el bit 2 (WDTE) de la Palabra de Configuración.

En el registro STATUS existe un bit denominado /TO que pasa a valer 0 después del desbordamiento del WDT.

TABLE 6-7: SUMMARY OF REGISTERS ASSOCIATED WITH THE WATCHDOG TIMER

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
2007h	Config. bits	(2)	(2)	(2)	(2)	$\overline{\text{PWRTE}}$(1)	WDTE	FOSC1	FOSC0	(2)	
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown. Shaded cells are not used by the WDT.
Note 1: See Register 6-1 for operation of the $\overline{\text{PWRTE}}$ bit.
2: See Register 6-1 and Section 6.12 for operation of the code and data protection bits.

Figura 4. *Registros asociados al sistema Watchdog.*

2.3. El temporizador TMR1

Algunos modelos de la gama media (p.ej. el 16F87X), además de disponer del temporizador TMR0 poseen otros dos llamados TMR1 y TMR2.

El TMR1 se trata de un temporizador/contador ascendente de 16 bits, por lo que está implementado mediante dos registros específicos TMR1H y TMR1L, que contienen el valor del contaje en cada momento. El valor del registro TMR1H-TMR1L evoluciona desde 0000h hasta FFFFh, en cuyo instante activa el señalizador TMR1IF (del registro PIR1) y vuelve a 0000h. Como fuente de los impulsos de reloj existen tres alternativas:

1. Generación interna (4·Tosc)

2. Generación mediante un oscilador externo controlado por cristal que se conecta a las patitas RC0/T1OSO/TICKI y RC1/T1OS1/CCP2. El oscilador se activa poniendo a 1 el bit T1OSCEN del registro T1CON.

   El bit TMR1CS del registro T1CON selecciona entre reloj interno o externo.

3. Trabaja en modo contador de eventos, cuando los impulsos externos a contar se aplican a la patita RC0/T1OSO/TICKI.

REGISTER 6-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)

bit 2 **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON**: Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figura 5*. Registro T1CON encargado del control del TMR1.*

TABLE 6-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

Figura 6. *Registros asociados al sistema Timer1.*

2.4. El temporizador TMR2

El TMR2 sólo está incorporado en unos pocos modelos de la gama media porque se trata de un temporizador de 8 bits diseñado para usarse conjuntamente con el circuito de modulación de anchura de impulsos (PWM).

REGISTER 7-1: T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
0010 = 1:3 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figura 7. *Registro T2CON encargado del control del TMR2.*

FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP

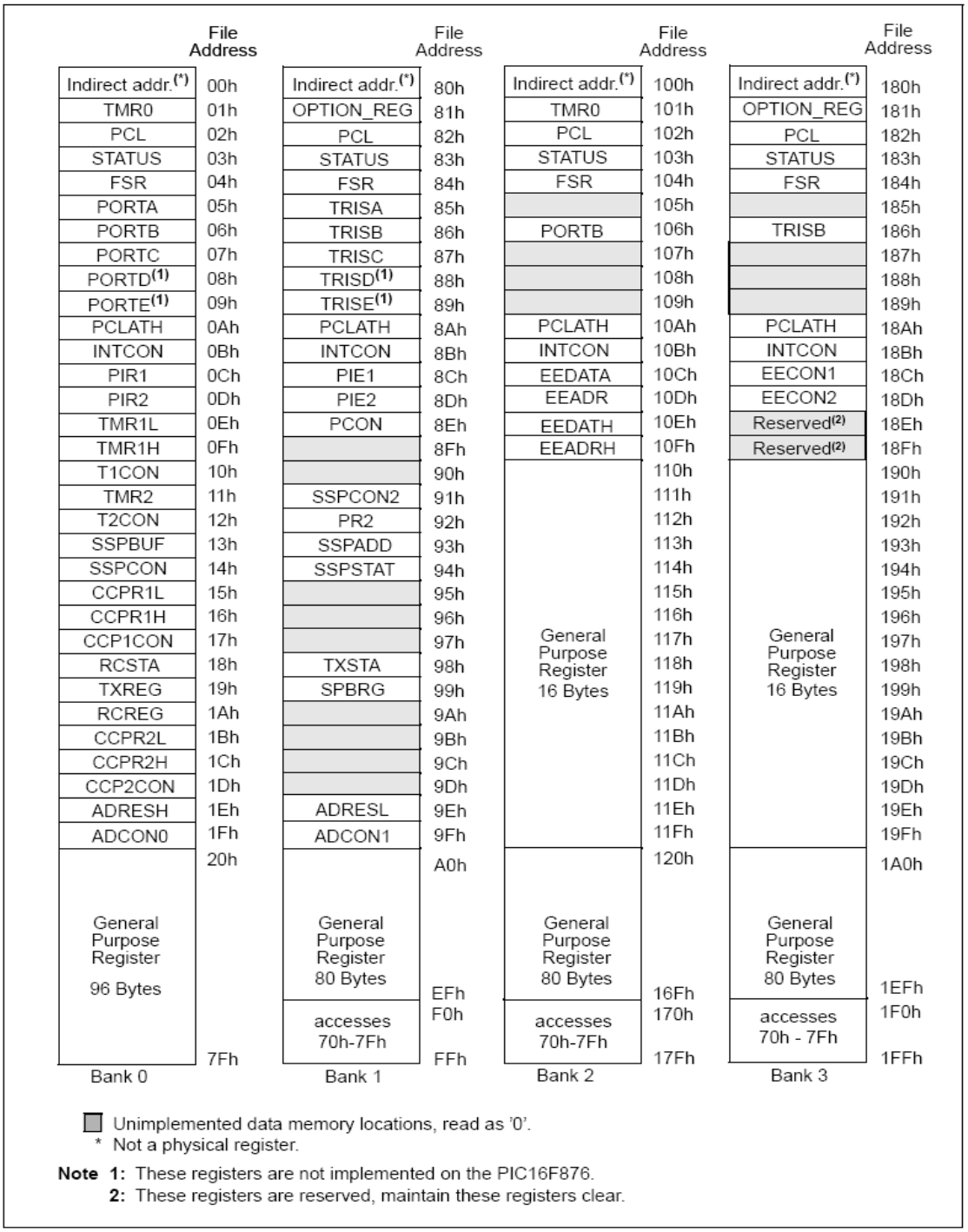


Figura 8. *Memoria de datos en el PIC16F877A.*

2.5. Displays 7 segmentos en la tarjeta EasyPIC

EasyPIC tiene 4 displays 7 segmentos en modo multiplexado. Las líneas de datos que conectan al microcontrolador con los displays son los bits de **PORTD** y cada display se habilita a través de los cuatro bit menos significativos de **PORTA**, es decir, RA0 = 1, activa el display DIS0, RA1=1 activa DIS1, RA2=1 activa DIS2 y RA3=1 activa DIS3, mientras que el dato a visualizar en cada display se envía a través de las líneas de PORTD.

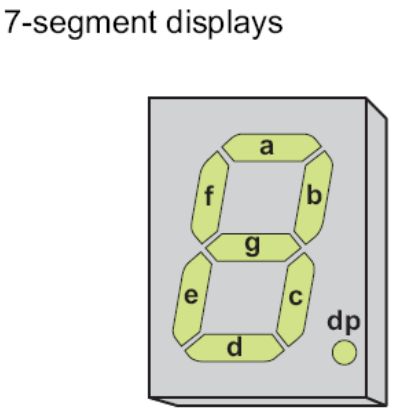


Figura 9. *Display 7 segmentos*.

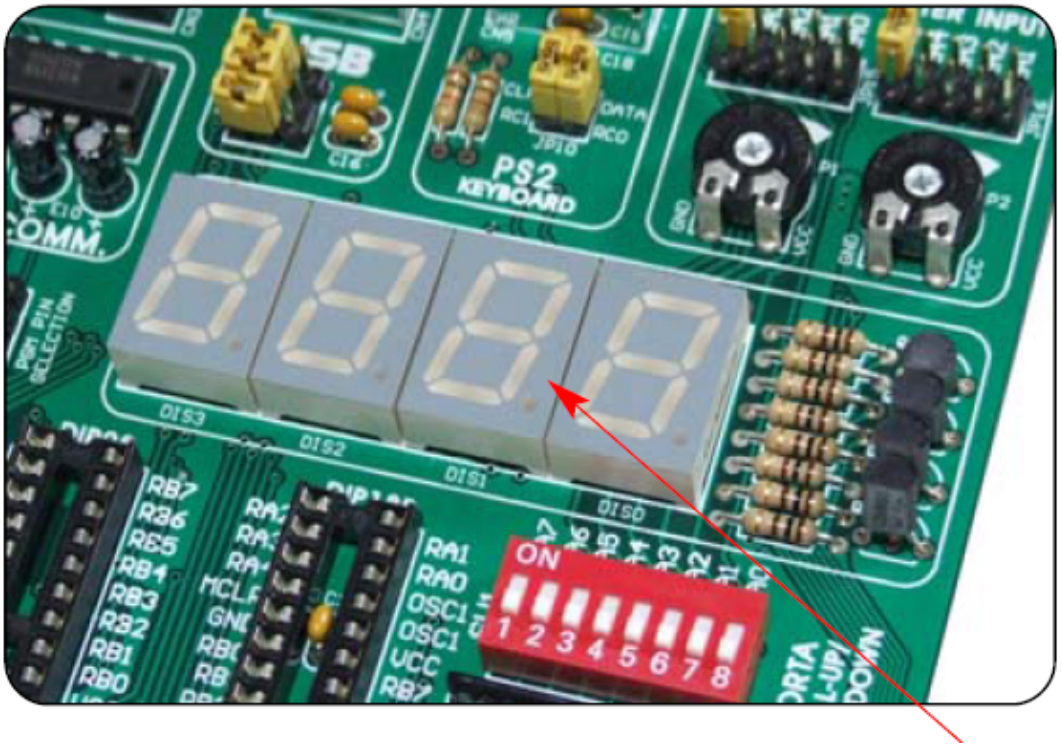


Figura 10. *EasyPIC tiene 4 displays 7 segmentos*.

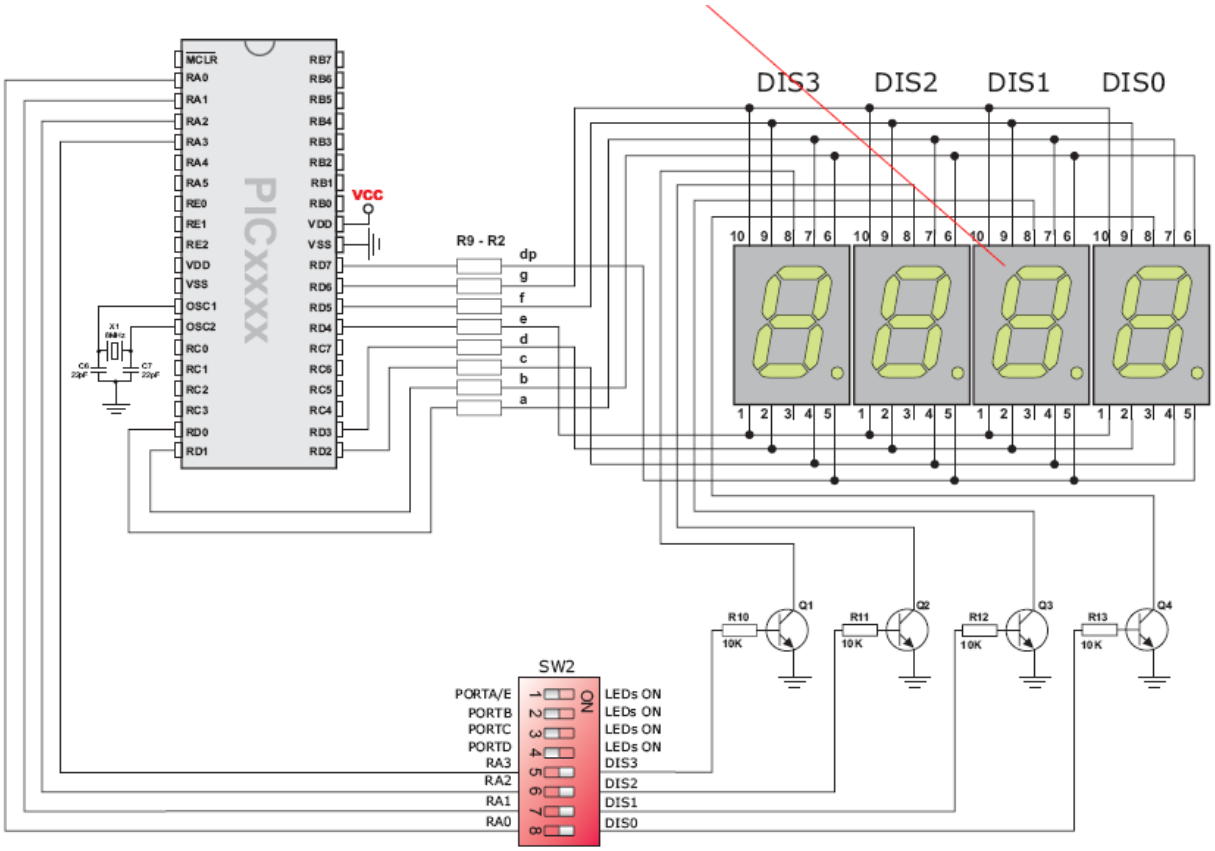


Figura 11. *Conexionado de los 4 displays 7 segmentos en la EasyPIC.*

3. EJERCICIOS

3.1. USO DEL TMR0 como temporizador

EJERCICIO 1:

a) Crea un proyecto en MPLAB y simula el funcionamiento del programa **Timer0_02.ASM.**

b) Utilizando la herramienta *StopWatch* del *Debugger*, mide el número de ciclos máquina y la duración en segundos de la subrutina Timer0_500us si el oscilador es de 4 MHz.

c) Cambia la frecuencia del oscilador a 8MHz (en *Debugger>>Settings*). ¿Cuantós ciclos máquina necesita la subrutina Timer0_500us? ¿Cuál es su duración en segundos?

EJERCICIO 2:

a) Crea un proyecto en MPLAB y simula el funcionamiento del programa **Timer0_03.ASM.**

b) Utilizando la herramienta *StopWatch* del *Debugger*, mide los ciclos máquina y la duración en segundos durante el periodo de tiempo que el bit RB3 está a nivel alto (utilizad un oscilador de 4 MHz).

c) Utilizando la herramienta *StopWatch* del *Debugger*, mide los ciclos máquina y la duración en segundos durante el periodo de tiempo que el bit RB3 está a nivel bajo (utilizad un oscilador de 4 MHz).

EJERCICIO 3:

a) Modifica el programa **Timer0_02.ASM** para que la subrutina de retardo cuente unos 30ms aproximadamente (oscilador de 8 MHz)**.**

b) Comprueba el funcionamiento de tu programa en la tarjeta EasyPIC. Utiliza el PIC16F84A.

EJERCICIO 4:

a) Modifica el programa **Timer0_02.ASM** para que la subrutina de retardo cuente 1 segundo aproximadamente (oscilador de 8 MHz). Para ello añade un bucle externo con un contador a la subrutina de temporización del TMR0, puesto que incluso con el mayor *prescaler* (256), el TMR0 por sí solo no puede contar ese tiempo. Por ejemplo, que el TMR0 cuente 5ms y el contador llegue hasta 200. (200 · 5ms=1s)

b) Comprueba el funcionamiento de tu programa en la tarjeta EasyPIC. Utiliza el PIC16F84A.

3.2. USO DEL TMR0 como contador

EJERCICIO 5:

a) El programa **Timer0_contador.asm** comprueba el funcionamiento del Timer 0 como contador de los impulsos aplicados a la línea RA4/T0CKI, donde se ha conectado un pulsador. Cada vez que presiona el pulsador situado en la línea RA4 se incrementa el contenido del TMR0 y la cuenta se visualiza en binario en los LEDS conectados a PORTB.

b) Comprueba el funcionamiento del programa **Timer0_contador.asm** en la tarjeta EasyPIC. Utiliza el PIC16F84A.

EJERCICIO 6:

a) El programa **cuenta_display.asm** comprueba el funcionamiento del Timer 0 como contador de los impulsos aplicados a la línea RA4/T0CKI, donde se ha conectado un pulsador. Cada vez que presiona el pulsador situado en la línea RA4 se incrementa el contenido del TMR0 y la cuenta se visualiza en uno de los displays 7 segmentos de la tarjeta EasyPIC.

b) Comprueba el funcionamiento del programa **cuenta_display.asm** en la tarjeta EasyPIC. Utiliza el PIC16F84A.

3.3. USO DEL TMR1 como temporizador

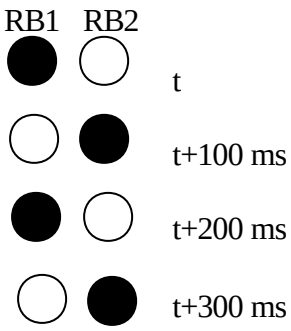
EJERCICIO 7:

a) El programa **Timer1.asm,** para el PIC16F877A, hace parpadear alternativamente los leds conectados a PORTC y PORTD con un periodo de 250ms entre el encendido y el apagado. El TMR1 se emplea para realizar la subrutina de retardo.

b) Comprueba el funcionamiento de **Timer1.asm** en la tarjeta EasyPIC. Utiliza el PIC16F877A.

EJERCICIO 8:

a) Confecciona un programa en ensamblador para el PIC16F877A, suponiendo que se ejecutará sobre la tarjeta EasyPIC4 (F_{OSC}=8Mhz), que produzca el apagado y encendido de los leds conectados a los bits 1 y 2 del puerto B (RB1 y RB2) cada 100 ms haciendo uso del sistema TIMER1. Nunca deben estar encendidos o apagados los dos leds simultáneamente, es decir, la secuencia debe ser:



Para el TIMER1 utiliza la siguiente fórmula para el cálculo del tiempo de temporización, puesto que el registro TMR1 consta de 16 bits:

Temporización=T_{CM} · Prescaler · (65536-Carga TMR1)

- ***Temporización*** es el tiempo deseado.
- ***T_{CM}*** es el período de un ciclo máquina. Para 8 MHZ, T_{CM}=0.5µs.
- ***Prescaler*** es el rango de divisor de frecuencia elegido.
- ***(65536-Carga TMR1)*** es el número total de impulsos a contar por el TMR1 antes de desbordarse en la cuenta ascendente.

b) Comprueba el funcionamiento de tu programa en la tarjeta EasyPIC. Utiliza el PIC16F877A.

EJERCICIO 9:

a) Confecciona un programa en ensamblador para el PIC16F877A, suponiendo que se ejecutará sobre la tarjeta EasyPIC4 (F_{OSC}=8Mhz), que haga una cuenta atrás de 9 a 0 y se visualice en el display 7 segmentos **DIS0** con una temporización de 1 segundo entre cada dígito. Utiliza el TIMER1 para la subrutina de retardo.

b) Comprueba el funcionamiento de tu programa en la tarjeta EasyPIC. Utiliza el PIC16F877A.

EJERCICIO OPCIONAL:

EJERCICIO 10:

a) Confecciona un programa en ensamblador para el PIC16F877A, suponiendo que se ejecutará sobre la tarjeta EasyPIC4 (F_{OSC}=8Mhz), que produzca la visualización de la palabra "HOLA" en los displays 7 segmentos de la tarjeta EasyPIC.

b) Comprueba el funcionamiento de tu programa en la tarjeta EasyPIC. Utiliza el PIC16F877A.

AYUDA:

Dígito H → 0x76

Dígito O → 0x3F

Dígito L → 0x38

Dígito A → 0x77