TECNOLOGÍA ELECTRÓNICA DE COMPUTADORES

2º Curso – GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

Tema 9: Circuitos integrados: microcontroladores

Lección 21. Microcontroladores PIC 16xxx: Temporizadores

Lección 21. Microcontroladores PIC 16xxx: Temporizadores

21.1. Características generales de los temporizadores

21.2. El temporizador Timer0 (TMR0). Características generales

21.3. Diagrama de bloques del Timer0

21.4. Configuración del Timer0 (TMR0) y del Watchdog (WDT)

21.5. Realización de temporizaciones: registro TMR0 y Flag T0IF

21.6. Ejemplo de uso del temporizador TMR0

21.7. Anexos

- **Retardos tras escribir en el TMR0. Uso del prescaler y retardos**
- **Uso de un reloj externo: condiciones de sincronización**
- **Cambios en la asignación del prescaler: precauciones**

21.1. Características generales de los temporizadores

• Definición: un temporizador es un dispositivo que marca o indica el transcurso de un tiempo determinado

•Los PIC16F87X tienen 3 módulos temporizadores denominados:

- TIMER0 (TMR0)
- TIMER1 (TMR1)
- TIMER2 (TMR2).

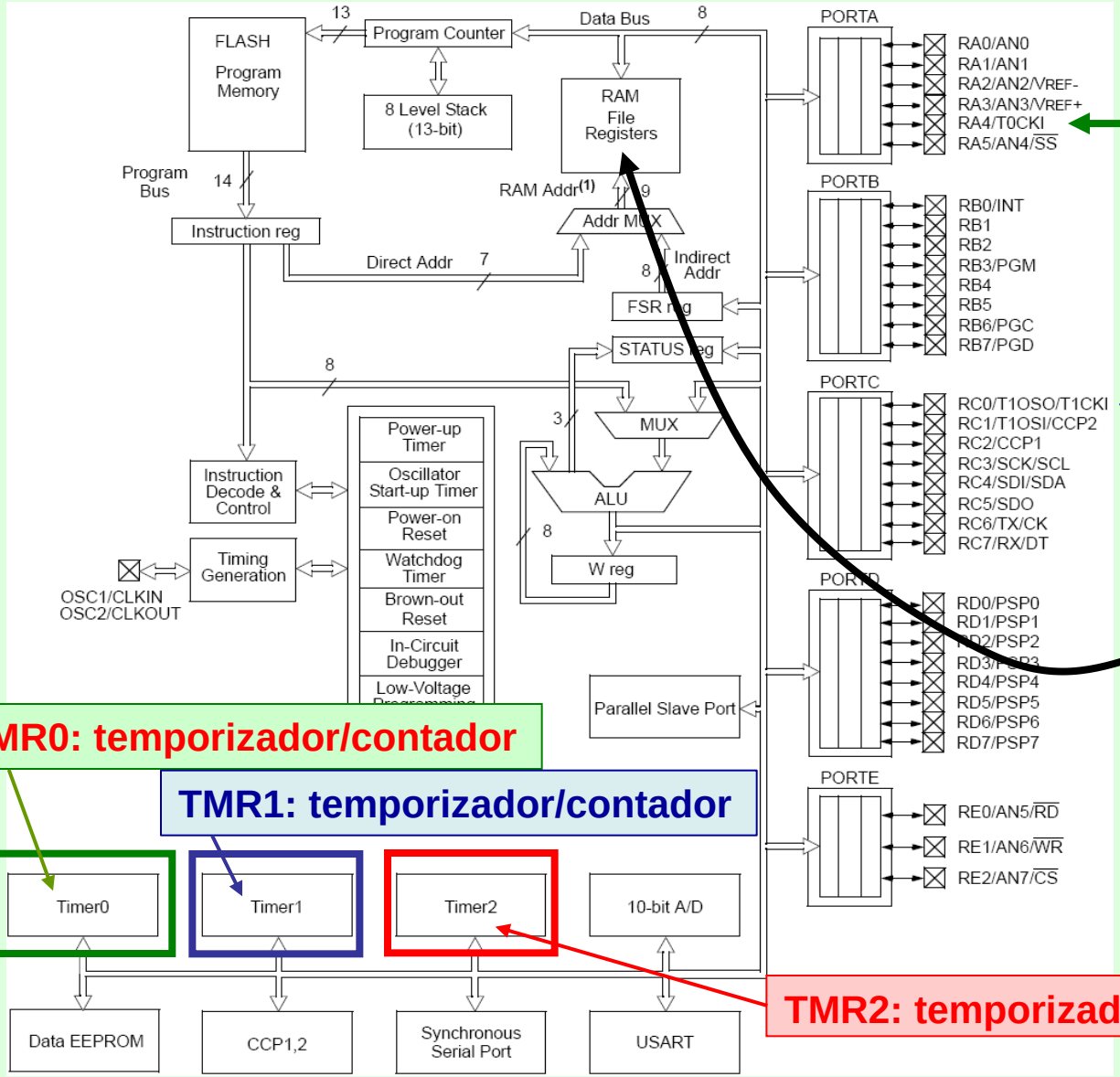
• Los módulos temporizadores en los microcontroladores PIC se emplean para:

- Contabilizar intervalos de tiempo
- Contar flancos que aparecen en pines externos del micro (sólo TMR0 y TMR1)

• Cuando trabajan como temporizadores, utilizan como patrón de cuenta un reloj que se genera a partir del oscilador del microcontrolador

• Cada módulo puede generar una interrupción para indicar algún evento si:

- Se ha sobrepasado el valor máximo de cuenta de un temporizador (*overflow*)
- Se ha alcanzado un valor dado



T0CKI
Entrada de flancos para TMR0 en modo contador

T1CKI
Entrada de flancos para TMR1 en modo contador

La configuración de los Temporizadores se realiza mediante la carga de ciertos registros presentes en RAM

TMR0: temporizador/contador

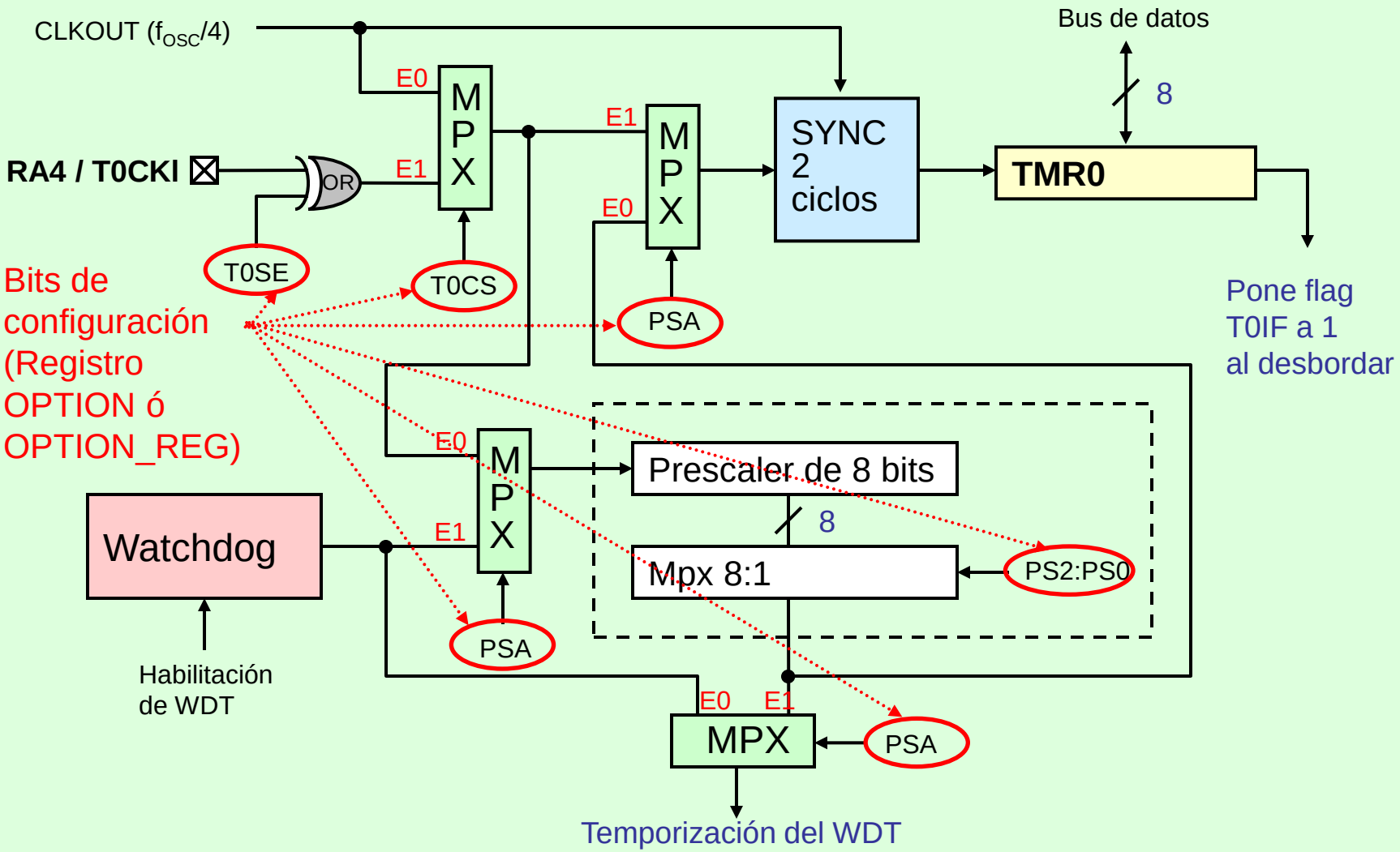
TMR1: temporizador/contador

TMR2: temporizador

21.2. Temporizador Timer0 (TMR0): características generales

- El TMR0 se basa en un contador ascendente de 8 bits al que se accede mediante un registro en RAM denominado TMR0 (posiciones 01h-101h).
- Dos modos de trabajo según la fuente del reloj:
  - Modo temporizador. Usa un reloj interno generado a partir del reloj del microcontrolador (fosc/4)
  - Modo contador. Usa un reloj externo que entra a través del pin RA4/T0CKI
- Puede utilizar un prescaler o divisor de frecuencia previo de 8 bits cuyo valor de división es configurable por software.
- Permite solicitar interrupciones cuando se produce un desbordamiento (*overflow*) del registro TMR0. Es decir cuando pasa del valor 0xFF al 0x00.
- Para el caso de cuenta de pulsos de un reloj externo, se puede seleccionar en qué flanco (de subida o de bajada) se realizará el incremento**.**
- El registro TMR0 es accesible desde el programa :
  - Se puede leer (ejemplo: movf TMR0,W)
  - Se puede escribir (ejemplo: movwf TMR0)

21.3.Diagrama de Bloques del TEMPORIZADOR TMR0



21.4. Configuración del TMR0 y del Watchdog (WDT)

- Los bits de configuración del TMR0 están en el registro OPTION
- En el fichero de inclusión de etiquetas de registros y bits P16F877.INC se denomina OPTION_REG para distinguirlo de la antigua instrucción OPTION

Registro OPTION_REG

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Asignación del prescaler

- El divisor de frecuencia interno se asigna bien al TMR0 ó bien al Watchdog (WDT) mediante el bit PSA:
  - ➢ Si PSA=1, entonces el prescaler es utilizado por el WDT y TMR0 contabiliza directamente los flancos sin división alguna
  - ➢ Si PSA=0, entonces el prescaler es utilizado por el módulo TIMER0

Bits de configuración del Timer0 y Watchdog

bit 7 **$\overline{\text{RBPU}}$**

bit 6 **INTEDG**

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
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

Configuración del TMR0

Registro OPTION_REG

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

El bit T0CS (OPTION_REG<T0CS>) selecciona la fuente de reloj para el TIMER0:

- Si T0CS=0, el TMR0 cuenta flancos a partir del reloj interno (frecuencia base fosc/4). Modo temporizador
- Si T0CS=1, el TMR0 cuenta tomando como base los flancos que entran al microcontrolador por el pin RA4/T0CKI. Modo contador

Si se cuentan pulsos del pin RA4/T0CKI, el bit T0SE (OPTION_REG<T0CKI>) permite seleccionar el flanco de la señal en el que se produce el incremento de la cuenta (entrada de la puerta EXOR)

- Si T0SE=0, se selecciona el flanco de subida.
- Si T0SE=1, se selecciona el flanco de bajada.

Uso del prescaler (divisor de frecuencia)

Registro OPTION_REG

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

- Si se quiere utilizar el prescaler con TMR0 hay que ajustar: OPTION_REG<PSA>=0 (Asigna prescaler al TMR0)

- El Timer0 sólo se incrementa cada “n” flancos de reloj (interno o externo) es decir el prescaler trabaja como un divisor de frecuencia por n

- El valor del prescaler ”n” viene definido por el valor de los bits PS2:PS0 (OPTION_REG<2:0>) de acuerdo a la tabla

Bit Value	TMR0 Rate	WDT Rate
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

21.5. Realización de temporizaciones: el TMR0 y flag T0IF

• Para realizar una temporización, se cargará en el TMR0 el valor adecuado, en función del tiempo que se desea temporizar

• Luego se espera al rebosamiento (overflow) del TMR0: al pasar de 0xFF a 0x00 se pone a 1 el flag T0IF (INTCON<2>)

• El programa puede detectar en este flag que ha terminado la temporización.

Puede hacerse de dos formas:

a) Mediante un programa que compruebe este FLAG periódicamente

b) Generando una interrupción (que es lo usual) al producirse el desbordamiento

Cálculo de la temporización en TMR0

$$\text{Temp TMR0} = [(256 - \text{carga}) \cdot \text{PS} + 2] \cdot T_{instr}$$

- Temporización
- Incrementos hasta desbordamiento
- PS es el Prescaler, toma el valor 1 si se asigna el prescaler al Watchdog
- Hay 2 ciclos de sincronización tras cargar TMR0, lo que introduce un retardo adicional
- $4 / f_{osc}$

• Se llama **carga** al valor que se asigna al registro TMR0 (01h, 101h) al comenzar la temporización.

• El valor del *prescaler* **PS** queda determinado por los tres bits más bajos del registro OPTION_REG (81h, 181h). Si el *prescaler* se asigna al *watchdog* (bit PSA=1), se tiene **PS**=1 para la anterior expresión ya que no hay división de frecuencia.

TEMPORIZADOR TMR0 - Interrupciones

• La interrupción del Timer0 se origina al producirse el rebosamiento (*overflow*) de TMR0, al pasar de 0xFF a 0x00: entonces se pone a 1 el flag T0IF (INTCON<2>).

• Si la interrupción de TMR0 está activada:

- ➢ Se detiene la ejecución de la siguiente instrucción
- ➢ Se salta a la rutina de interrupción (posición 0x0004 de la memoria de programa)

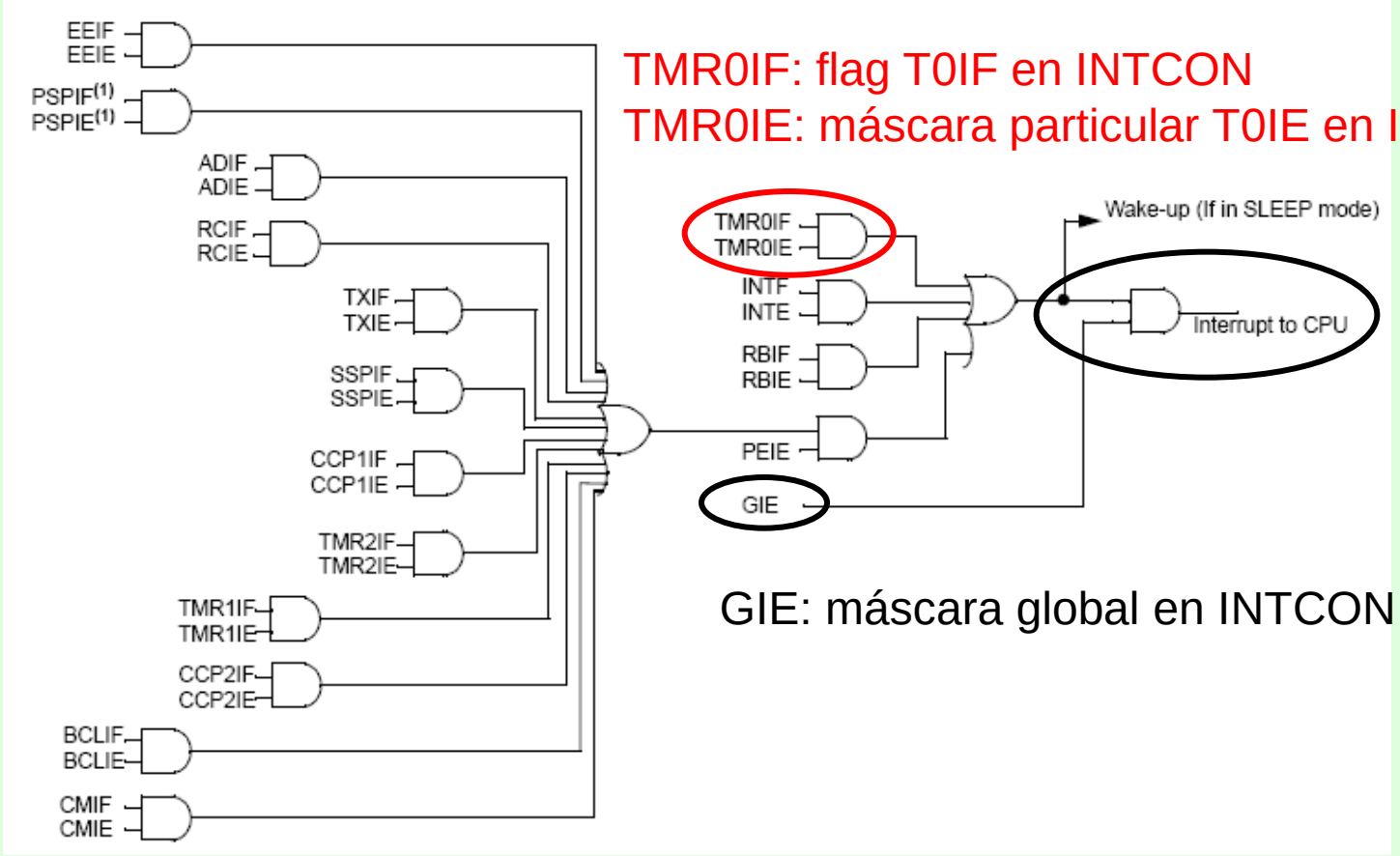
• Condiciones para que esté activada:

- ➢ El bit de enmascaramiento particular T0IE (INTCON<5>) debe estar a "1" (Interrupción de TMR0 activa)
- ➢ La máscara global de interrupciones GIE (INTCON<7>) debe estar a "1"

• Antes de salir de la rutina de interrupción del TMR0 (con retfie) debe limpiarse el flag T0IF (con bcf INTCON,T0IF por ejemplo) ya que si no se produciría una nueva entrada en la interrupción.

NOTA: La interrupción del TMR0 no puede despertar al microcontrolador del modo Sleep (dormido), ya que el TIMER0 está apagado durante este modo.

Lógica de interrupciones: detalle para TMR0



TMR0IF: flag T0IF en INTCON
TMR0IE: máscara particular T0IE en INTCON

GIE: máscara global en INTCON

0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

TEMPORIZADOR TMR0 – Registros asociados al módulo TMR0

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
01h, 101h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE(1)	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h, 181h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	PORTA Data Direction Register(1)						--11 1111	--11 1111

Comentarios:

• El pin RA4/T0CKI debe estar definido como entrada si se utiliza como fuente de reloj para el TMR0 el reloj externo que "entra" por ese pin.

• Todas las instrucciones de escritura sobre el registro TMR0 (CLRF TMR0, MOVWF TMR0, BSF TMR0,bitx, etc.) realizan una limpieza del prescaler.

21.6. Ejemplo de uso del temporizador TMR0

• Se pretende realizar una intermitencia sobre un led controlado desde el pin 3 del PORTB en la placa PICDEM2-PLUS de manera que permanezca 0,5 s encendido y 0,5 s apagado. Si la salida está a 1 luce y si está a 0 se apaga.

• La temporización se va a realizar mediante el temporizador TMR0, de manera que, tras realizar una carga del mismo el desbordamiento se produzca a los 500ms. De acuerdo con la anterior fórmula, la temporización máxima que se podría realizar con un oscilador de 4MHz sería:

Tmax = [(256-0)*256 + 2] * 4/4MHz = 65.538 µs = 65,538 ms < 500 ms

lo que no permite alcanzar el tiempo total a temporizar, por tal motivo se emplea un contador que acumule temporizaciones menores hasta alcanzar los 500 ms, por ejemplo se pueden realizar temporizaciones de 50 ms y contabilizar un total de 10. En ese caso, se debe cumplir:

50 ms = [(256-Precarga)*256+2] * 4/4MHz

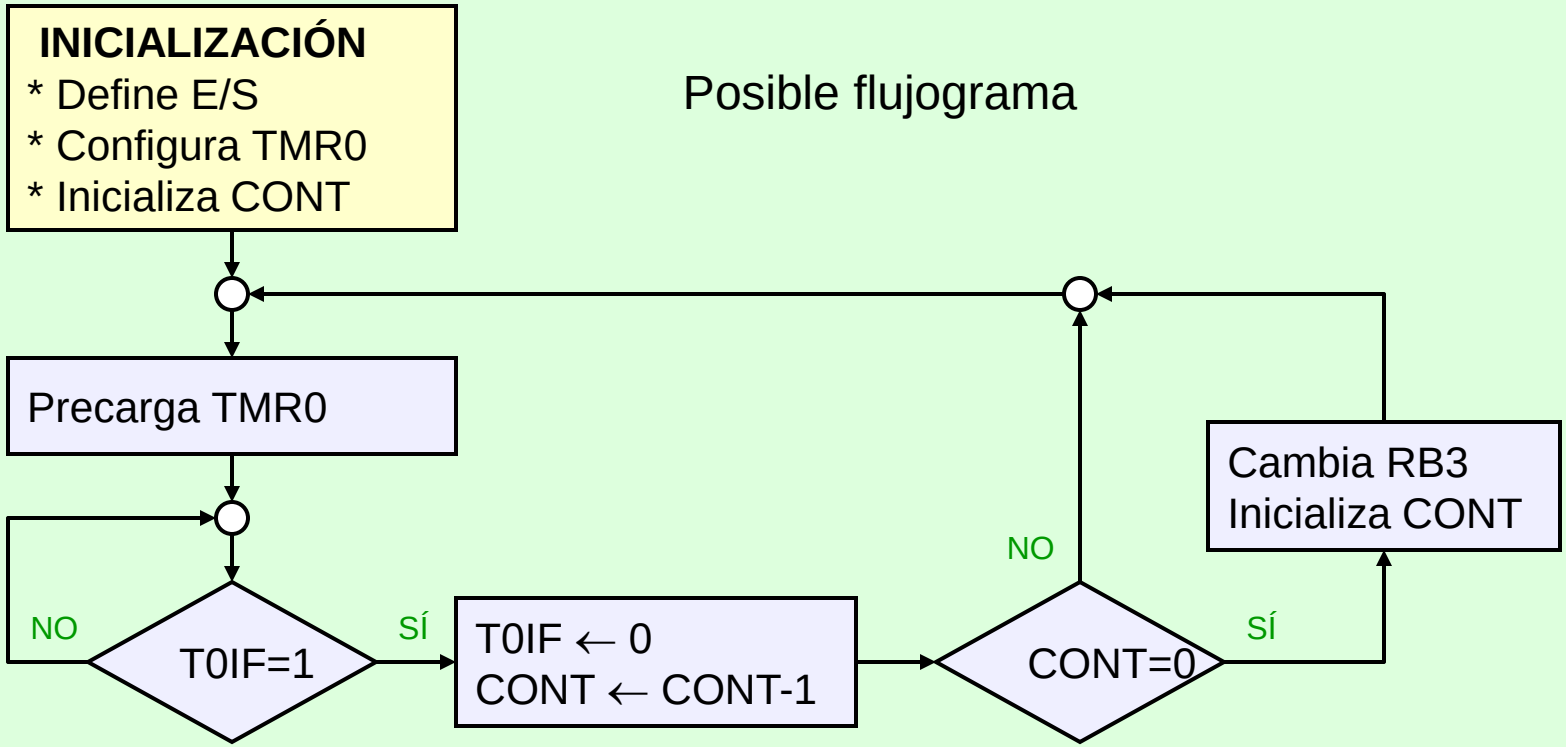
y despejando en la anterior expresión: Precarga= 60,69 -> 60 (aprox. entera)

Con lo que resulta: T= [(256-60)*256+2] * 4/4MHz = 50,178 (ms)

Solución al ejemplo 1 (temp01.asm)

• Utiliza el temporizador TMR0 para hacer que el LED conectado a RB3 parpadee: 500ms encendido y 500ms apagado.

Posible flujograma



```
;********************************* temp01.asm *********************************
;
; El LED conectado al bit 3 del Puerto B parpadea de modo que está 500ms encendi-
; do y otros 500ms apagado. Se usa el temporizador TMR0 para establecer la tempo-
; rización. (Se considera que el oscilador del PIC es de 4MHz).

; ZONA DE DATOS ******************************************************************
;       Configuración para el grabador
              __CONFIG   _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_ON & _LVP_OFF

              LIST       P=16F877  ; Procesador.
              INCLUDE <P16F877.INC>          ; Definición de los operandos utilizados.

CONT          EQU        0x20                ; Cuenta las veces que se desborda TMR0.

; ZONA DE CÓDIGOS ****************************************************************
              ORG        0                   ; El programa comienza en la dirección 0 de
                                             ; memoria de programa.
Inicio        bsf        STATUS,RP0          ; Pone a 1 el bit 5 de STATUS. Acceso al Banco 1.

              movlw      b'11110111‘         ; Se configura el bit 3 de PORTB como salida,
              movwf      TRISB               ; el resto de bits queda como entradas.

              movlw      0x07                ; Prepara TMR0 para contar pulsos de oscilador y
              movwf      OPTION_REG          ; le asigna un prescaler de 256.
```

```
            bcf     STATUS,RP0      ; Pone a 0 el bit 5 de STATUS. Acceso al Banco 0.

            clrf    PORTB           ; Inicializa PORTB a 0.
            movlw   0x0A            ; Inicializa la variable
            movwf   CONT            ; CONT a 10.

;    Con un oscilador de 4MHz, la máxima temporización que se alcanza con TMR0 es
; de 65,5ms. Para poder llegar a temporizar los 500ms hace falta que TMR0 se
; desborde varias veces. Por ello se preparará TMR0 para temporizar 50ms y se
; esperará que se produzca esta temporización 10 veces.

Principal   movlw   d'60'           ; Carga el valor 60 en el registro TMR0 (con este
            movwf   TMR0            ; valor se temporizan 50ms).
Espera      btfss   INTCON,T0IF     ; Espera que termine la temporización, lo cual se
            goto    Espera          ; detecta cuando el flag T0IF se pone a 1.
            bcf     INTCON,T0IF     ; Baja el flag.
            decfsz  CONT            ; Decrementa CONT y si es 0 ignora la siguiente
instrucción
            goto    Principal       ; Si sigue siendo distinto de 0 lanza una nueva
temporización.
            comf    PORTB,F         ; Si CONT=0, cambia el valor de RB3,
            movlw   0x0A            ; y reinicializa la variable CONT
            movwf   CONT;           ; dándole de nuevo el valor CONT=10
            goto    Principal       ; antes de lanzar una nueva temporización.

            END                     ; Fin del programa.
```

21.6. Anexos

Temporizador TMR0: Retardos tras escribir TMR0

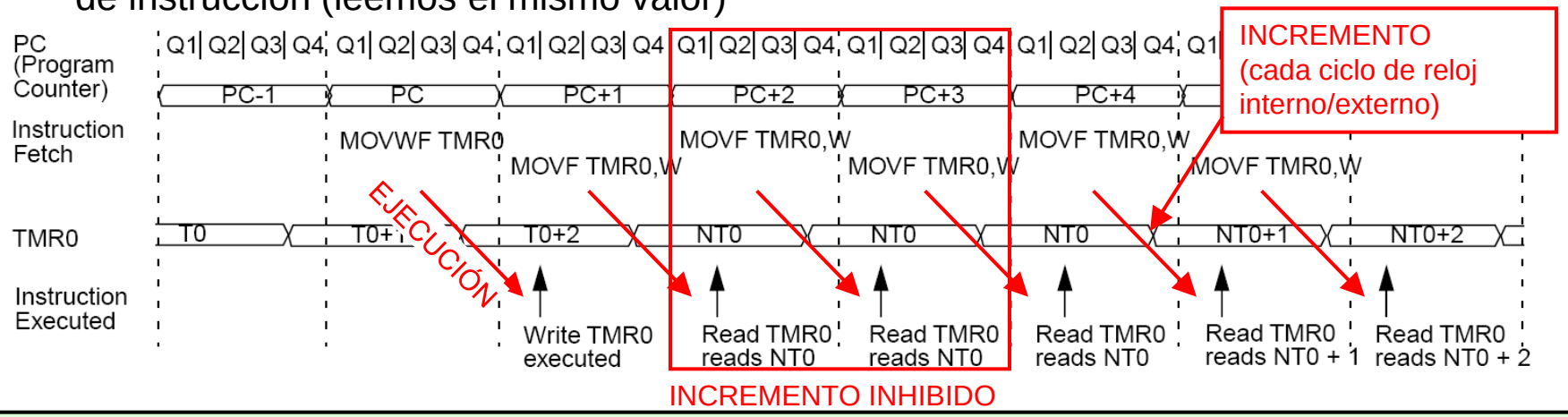
Al cargar un valor en el registro TMR0 (se escribe mediante una instrucción), se produce un retardo de dos ciclos de instrucción durante los cuales se inhibe tanto el prescaler como TMR0.

Es necesario tener en cuenta esa inhibición temporal al realizar una precarga (y sumar los ciclos de instrucción "perdidos" para calcular el valor de temporización correcto)

EJEMPLO DE CUENTA DE TMR0:
- Sin prescaler (PSA=1)
- Con fuente de reloj interna (T0CS=0)

Escribimos el registro TMR0 y luego lo leemos para ver cuando cambia. Observar que:

- El valor de TMR0 se lee al acabar Q1 y se actualiza al acabar Q3 del ciclo de instrucción
- Tras la escritura en el registro TMR0, el incremento se inhibe durante los dos siguientes ciclos de instrucción (leemos el mismo valor)



Uso del prescaler (divisor de frecuencia) y retardos

- Si se quiere utilizar el prescaler con TMR0 hay que ajustar: OPTION_REG<PSA>=0
- El Timer0 sólo se incrementa cada “n” flancos de reloj (interno o externo)
- El valor del prescaler ”n” viene definido por el valor de los bits PS2:PS0 (OPTION_REG<2:0>) de acuerdo a la tabla

Bit Value	TMR0 Rate	WDT Rate
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

EJEMPLO DE CUENTA DE TMR0:

- Con prescaler (PSA=0)
- Con un valor “n” de prescaler 1:2 (PS<2:0>=000b)
- Con fuente de reloj interna (T0CS=0)

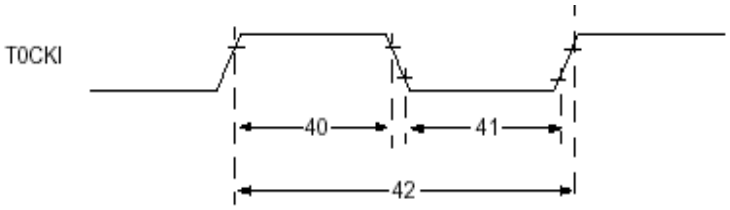
Observar de nuevo que, tras la escritura en el registro TMR0, el incremento se inhibe durante los dos siguientes ciclos de instrucción (leemos el mismo valor)

Temporizador TMR0 – Uso del reloj externo (Sincronización)

• Cuando se usa el reloj externo, se deben cumplir ciertos requisitos para asegurar que el reloj externo pueda sincronizarse con el reloj interno (Tosc), y puede existir un retardo entre el flanco en el pin RA4/T0CKI y el incremento real del TIMER0.

• Requisitos genéricos de la señal de reloj externa:

- Tiempos mínimos a nivel alto y a nivel bajo
- Los tiempos mínimos dependen de si se usa o no prescaler
- El límite absoluto viene dado por un valor mínimo de 10 ns para el ancho de pulso.
- Consultar las hojas de características en cada caso



Param No.	Symbol	Characteristic		Min	Typ †	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
42	Tt0P	T0CKI Period		GREATER OF: 20 µS OR TCY + 40 / N	—	—	ns	N = prescale value (1, 2, 4,..., 256)

Temporizador TMR0 – Cambio del Prescaler

- La asignación del prescaler al TMR0 (PSA=0) ó al WATCHDOG (PSA=1) puede realizarse en cualquier momento del programa.
- El cambio de la asignación del prescaler del TMR0 al Watchdog en medio de un programa puede provocar un RESET no deseado en el microcontrolador, debido al desbordamiento del Watchdog durante la ejecución del programa.
- El cambio de la asignación del prescaler de un módulo a otro es crítico y debe realizarse como se indica en los siguientes fragmentos de programas.
- Esta precaución debe tenerse en cuenta incluso si el WDT está deshabilitado.

;Cambio de prescaler del TIMER0 al WDT

```
BSF STATUS, RP0          ;Banco 1
MOVLW b'xx0x0xxx'        ;Selección de fuente de reloj y valor
MOVWF OPTION_REG         ;del prescaler distinto al 1:1
BCF STATUS, RP0          ;Banco 0
CLRF TMR0   ;Limpio TMR0 y prescaler antes
BSF STATUS, RP1          ;Banco 1
MOVLW b'xxxx1xxx'        ;Asigno al WDT, no se cambia
MOVWF OPTION_REG         ;valor del prescaler
CLRWDT                   ;Limpio WDT y prescaler
MOVLW b'xxxx1xxx'        ;Selección del nuevo valor
MOVWF OPTION_REG         ;del prescaler y asigna al WDT
BCF STATUS, RP0          ;Banco 0
; Lineas 2 y 3 del programa no tienen que ser incluidas si el valor
;del prescaler deseado es distinto a 1:1.
;Si 1:1 es el valor final deseado, entonces un valor de prescaler
;temporal se establece en las líneas 2 y 3 y el valor final
;del prescaler en las líneas 10 y 11.
```

;Cambio de prescaler del WDT al TIMER0

```
CLRWDT                   ;Limpia WDT y prescaler
BSF STATUS, RP0          ;Banco 1
MOVLW b'xxxx0xxx‘        ;Selecciono TMR0, nuevo valor
                         ;del prescaler y fuente de reloj
MOVWF OPTION_REG ;
BCF STATUS, RP0          ;Banco 0
```

OPTION_REG

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

PSA=1 (WDT), PSA=0 (TMR0)
T0CS=0 (CLK_int) T0CS=1 (CLK_ext)