SISTEMAS ELECTRÓNICOS Y AUTOMÁTICOS
PRACTICAS DE MICROCONTROLADORES PIC

PRÁCTICA 10:

INTERRUPCIONES (II)

- ***Ejercicios y ejemplos de la interrupción por desbordamiento del TIMER0.***

---

1. Interrupción por desbordamiento del TIMER0

Para autorizar la interrupción por desbordamiento del TMR0 los bits T0IE y GIE del registro INTCON deben posicionarse a "1". En estas condiciones cuando el temporizador TMR0 se desborda, al pasar de b'11111111' (FFh) a b'00000000' (00h), activa el flag T0IF del registro INTCON produciendo una interrupción.

En la práctica 5 vimos que para calcular los tiempos a controlar con el sistema Timer0 se utiliza la siguiente fórmula:

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.

A continuación se expone el código de un programa ejemplo (***Int_T0I_01.asm***), donde se explica una forma sencilla de generar ondas cuadradas, que en caso de conectar un altavoz al pin RB3 produciría un pitido.

Para una reloj de 4 MHz, la onda cuadrada producida tiene una frecuencia de 10kHz (cada semiperiodo dura 50 microsegundos). Estos tiempos de temporización se logran mediante la interrupción por desbordamiento del TIMER 0.

EJERCICIO 1:

a) Crea un proyecto en MPLAB y simula el funcionamiento del programa **Int_T0I_01.asm.**

b) Utilizando la herramienta *StopWatch* del *Debugger*, mide la duración de los semiperiodos de la onda cuadrada para el nivel alto y el nivel bajo.

```
;**********************************Int_T0I_01.asm ***********************************
;
;       ======================================================================
;         Del libro "MICROCONTROLADOR PIC16F84. DESARROLLO DE PROYECTOS"
;         E. Palacios, F. Remiro y L. López.
;         Editorial Ra-Ma.  www.ra-ma.es
;       ======================================================================
;
; Por la línea 3 del puerto B se genera una onda cuadrada de 10 kHz, cada semiperiodo dura
; 50 µs. Los tiempos de temporización se lograrán mediante la interrupción por desbordamiento
; del Timer 0. A la línea de salida se puede conectar un altavoz que producirá un pitido.
;
; ZONA DE DATOS ********************************************************************

        __CONFIG   _CP_OFF &  _WDT_OFF & _PWRTE_ON & _XT_OSC
        LIST       P=16F84A
        INCLUDE  <P16F84A.INC>

        CBLOCK  0x0C
        ENDC

TMR0_Carga50us   EQU      -d'50'
#DEFINE          Salida   PORTB,3

; ZONA DE CÓDIGOS ******************************************************************

        ORG     0
        goto    Inicio
        ORG     4                       ; Vector de interrupción.
        goto    Timer0_Interrupcion
Inicio
        bsf     STATUS,RP0              ; Acceso al Banco 1.
        bcf     Salida                  ; Línea configurada como salida.
        movlw   b'00001000'
        movwf   OPTION_REG              ; Sin prescaler para TMR0 (se asigna al Watchdog).
        bcf     STATUS,RP0              ; Acceso al Banco 0.
        movlw   TMR0_Carga50us          ; Carga el TMR0.
        movwf   TMR0
        movlw   b'10100000'
        movwf   INTCON                  ; Autoriza interrupción T0I y la general (GIE).
Principal                                       ; No puede pasar a modo de bajo consumo
        goto    $                       ; porque detendría el Timer 0.

; Subrutina "Timer0_Interrupcion" --------------------------------------------------
;
; Como el PIC trabaja a una frecuencia de 4 MHz el TMR0 evoluciona cada microsegundo. Para
; conseguir un retardo de 50 µs con un prescaler de 1 el TMR0 tiene que contar 50 impulsos.
; Efectivamente: 1 µs x 50 x 1 = 50 µs.
;
Timer0_Interrupcion
        movlw   TMR0_Carga50us
        movwf   TMR0                    ; Recarga el timer TMR0.
        btfsc   Salida                  ; Testea el anterior estado de la salida.
        goto    EstabaAlto
EstabaBajo
        bsf     Salida                  ; Estaba bajo y lo pasa a alto.
        goto    FinInterrupcion
EstabaAlto
        bcf     Salida                  ; Estaba alto y lo pasa a bajo.
FinInterrupcion
        bcf     INTCON,T0IF             ; Repone flag del TMR0.
        retfie

; Comprobando con el simulador del MPLAB se obtienen unos tiempos para la onda cuadrada de
; 56 µs para el nivel alto y de 55 µs para el bajo.

        END
```

2. TEMPORIZACIONES EXACTAS

En caso de requerir temporizaciones exactas, hay que tener en cuenta el tiempo de ejecución de las instrucciones y de los saltos. El valor de carga del TMR0 será algo menor y habrá que hacer un ajuste fino con instrucciones ***nop***.

Esto normalmente se realiza experimentalmente utilizando el simulador MPLAB y el reloj de la ventana *(StopWatch) del Debugger.*

El programa ejemplo ***Int_T0I_02.asm*** genera una señal cuadrada de 10kHz, pero en esta ocasión calculada con exactitud.

EJERCICIO 2:

a) Crea un proyecto en MPLAB y simula el funcionamiento del programa **Int_T0I_02.asm.**

b) Utilizando la herramienta *StopWatch* del *Debugger*, mide la duración de los semiperiodos de la onda cuadrada para el nivel alto y el nivel bajo.

```
;********************************** Int_T0I_02.asm ***********************************
;
;       ================================================================
;         Del libro "MICROCONTROLADOR PIC16F84. DESARROLLO DE PROYECTOS"
;         E. Palacios, F. Remiro y L. López.
;         Editorial Ra-Ma.  www.ra-ma.es
;       ================================================================
;
; Por la línea 3 del puerto B se genera una onda cuadrada de 10 kHz. Cada semiperiodo dura
; 50 µs exactos. Los tiempos de temporización se lograrán mediante la interrupción del
; Timer 0. A la línea de salida se puede conectar un altavoz que producirá un pitido.
;
; El cálculo de la carga del TMR0 se realiza teniendo en cuenta los tiempos que tardan en
; ejecutarse las instrucciones y saltos para conseguir tiempos exactos. Para calcular el valor
; de carga del TMR0 se ayuda del simulador del MPLAB y de la ventana de reloj Stopwatch.
;
; ZONA DE DATOS *****************************************************************

        __CONFIG   _CP_OFF &  _WDT_OFF & _PWRTE_ON & _XT_OSC
        LIST       P=16F84A
        INCLUDE  <P16F84A.INC>

        CBLOCK  0x0C
        ENDC

TMR0_Carga50us   EQU     -d'43'          ; Obtenido experimentalmente con ayuda del
#DEFINE  Salida  PORTB,3                 ; simulador del MPLAB.

; ZONA DE CÓDIGOS ***************************************************************

        ORG     0
        goto    Inicio
        ORG     4                       ; Vector de interrupción.
        goto    Timer0_Interrupcion
Inicio
        bsf     STATUS,RP0              ; Acceso al Banco 1.
        bcf     Salida                  ; Esta línea se configura como salida.
        movlw   b'00001000'
        movwf   OPTION_REG              ; Sin prescaler para TMR0 (se asigna al Watchdog).
        bcf     STATUS,RP0              ; Acceso al Banco 0.
        movlw   TMR0_Carga50us          ; Carga el TMR0.
        movwf   TMR0
        movlw   b'10100000'
        movwf   INTCON                  ; Autoriza interrupción T0I y la general (GIE).
Principal                                       ; No puede pasar a modo de bajo consumo
        goto    $                       ; porque detendría el Timer 0.

; Subrutina "Timer0_Interrupcion" -----------------------------------------------------
;
; Como el PIC trabaja a una frecuencia de 4 MHz el TMR0 evoluciona cada microsegundo. Para
; conseguir un retardo de 50 microsegundos con un prescaler de 1 el TMR0 tiene que contar
; 43 impulsos. Efectivamente: 1µs x 1 x 43 + tiempo de los saltos y otros = 50 µs.
;
; Las instrucciones "nop" se ponen para ajustar el tiempo a 50 µs exacto y lograr una onda
; cuadrada perfecta. El simulador del MPLAB comprueba unos tiempos para la onda cuadrada de
; 10 kHz exactos de 50 µs para el nivel alto y otros 50 µs para el bajo.

Timer0_Interrupcion
        nop
        movlw   TMR0_Carga50us
        movwf   TMR0                    ; Recarga el TMR0.
        btfsc   Salida                  ; Testea el anterior estado de la salida.
        goto    EstabaAlto
EstabaBajo
        nop
        bsf     Salida                  ; Estaba bajo y lo pasa a alto.
        goto    FinInterrupcion
EstabaAlto
        bcf     Salida                  ; Estaba alto y lo pasa a bajo.
FinInterrupcion
        bcf     INTCON,T0IF             ; Repone flag del TMR0.
        retfie                          ; Retorno de interrupción

        END

;       ================================================================
;         Del libro "MICROCONTROLADOR PIC16F84. DESARROLLO DE PROYECTOS"
;         E. Palacios, F. Remiro y L. López.
;         Editorial Ra-Ma.  www.ra-ma.es
;       ================================================================
```

3. TEMPORIZACIONES LARGAS

Con la máxima división de frecuencia posible (prescaler=256) la temporización máxima que se puede conseguir para un circuito con cristal de cuarzo de 4MHz es para una (Carga TMR0=0) igual a:

$$Temporización = T_{CM} \cdot Prescaler \cdot (256 - Carga\ TMR0)$$
$$Temporización = 1 \cdot 256 \cdot (256 - 0) = 65536\ microsegundos = 65\ ms$$

Para lograr temporizaciones de tiempo mayores hay que utilizar un registro auxiliar de la forma que se explica en el programa ejemplo ***Int_T0I_03.asm***

EJERCICIO 3:

a) En el ejemplo **Int_T0I_03.ASM** un led conectado al bit 1 de PORTB (RB1, configurado como salida) se enciende durante 500ms y se apaga durante otros 500ms. Para conseguir la temporización de 500 ms, se repite 10 veces un lazo de 50 ms.

Modifica el programa **Int_T0I_03.ASM** para que funcione sobre la tarjeta EasyPIC con el microcontrolador PIC16F877A y que la intermitencia del LED sea de 500ms. Recuerda que en la tarjeta EasyPIC, el oscilador tiene una frecuencia de 8MHz, por lo tanto la interrupción del Timer0 ocurrirá cada 25 ms.

Cambios a realizar en el código:

	Int_T0I_03.ASM Según el libro	Int_T0I_03.ASM Para la EasyPIC
Modelo de microcontrolador:	**16F84A**	**16F877A**
Tipo de reloj	**XT**	**HS**
Posición de la variable *Registro50ms*	**0x0C**	**0x20**
Temporización	**F_{reloj}=4MHz** **500ms** **10 x 50ms**	**F_{reloj}=8MHz** **500ms** **20 x 25ms**

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

EJERCICIO 4:

a) En el ejemplo **Int_T0I_04.ASM** un led conectado al bit 1 de PORTB (RB1, configurado como salida) se enciende durante 800ms y se apaga durante otros 500ms.

   Modifica el programa **Int_T0I_04.ASM** para que funcione sobre la tarjeta EasyPIC con el microcontrolador PIC16F877A. Ten en cuenta los cambios a realizar que se detallan en la tabla del ejercicio 3.

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

EJERCICIO 5:

a) En el ejemplo **Int_T0I_05.ASM** se visualiza constantemente en el módulo LCD un mensaje largo que desplaza por la pantalla. Al mismo tiempo, un led conectado al bit 1 de PORTB (RB1, configurado como salida) se enciende durante 500ms y se apaga durante otros 500ms.

   Modifica el programa **Int_T0I_05.ASM** para que funcione sobre la tarjeta EasyPIC con el microcontrolador PIC16F877A. Ten en cuenta los cambios a realizar que se detallan en la tabla del ejercicio 3.

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

EJERCICIO 6:

a) **Int_Reloj_01.ASM** es un programa para un reloj digital en tiempo real sin puesta en hora. Se visualiza en el formato: "08:47:39". Las temporizaciones necesarias del reloj se logran mediante el Timer0, que produce una interrupción cada 50 ms.

   Modifica el programa **Int_ Reloj_01.ASM** para que funcione sobre la tarjeta EasyPIC con el microcontrolador PIC16F877A. Introduce sólo los cambios relativos al *Modelo de microcontrolador, Tipo de reloj y Posición de las variables.*

b) Comprueba el funcionamiento de tu programa en la tarjeta EasyPIC. El reloj funcionará más rápido (1 segundo de la pantalla equivale a 0.5 segundo reales) puesto que en la tarjeta EasyPIC el oscilador tiene una frecuencia de 8MHz.