Temporizaciones por Software

Temporizaciones:

En la mayor parte de los programas, resulta necesario controlar el tiempo que tardan en ejecutarse algunas acciones o bien establecer una duración determinada para un estado del sistema digital que se quiere realizar

¿Cómo se pueden realizar temporizaciones?

- Por software: manteniendo al microcontrolador ejecutando una zona de código de duración conocida y calculada (p.e. retardos mediante bucles)

- Por hardware: utilizando los módulos de temporización disponibles en el microcontrolador. Son contadores que cuentan a partir de un reloj externo o del propio oscilador del microcontrolador y que activan un indicador (flag) cuando desbordan o cuando alcanzan una determinada combinación

Diferencia entre temporizaciones software y hardware:

- Temporización software: el microcontrolador está ocupado ejecutando esa zona de código y no podría realizar otras acciones salvo que se activen interrupciones que podrían ser prioritarias ante la temporización en curso. Con el microcontrolador en modo de bajo consumo (SLEEP), no se podrían realizar este tipo de temporizaciones

- Temporización hardware: los módulos de temporización pueden realizar la "cuenta" de manera independiente al código que se esté ejecutando en ese momento.

  - Los flags que indican desbordamiento (o alcance), pueden provocar una interrupción si se configuran adecuadamente

  - Se pueden realizar varias temporizaciones de manera simultánea con los módulos disponibles, hasta 3 en el caso de la serie PIC16F87x (TMR0, TMR1 y TMR2)

  - El valor de temporización resulta ser más preciso

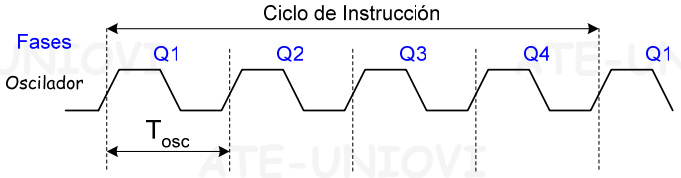
  - Si se cuentan flancos ajenos a los del oscilador, sería posible temporizar en modo de bajo consumo (TMR0 ó TMR1 como contadores)

Temporizaciones Software:

• Se debe evaluar la duración de la ejecución de una zona de código (contar ciclos de instrucción) destinada a la temporización

• PIC: arquitectura "pipeline" con ejecución y búsqueda simultánea de la siguiente instrucción presente en la memoria de programa. Todas las instrucciones se ejecutan en un ciclo de instrucción salvo las que impliquen la **no ejecución** de la instrucción que sigue a la que está en curso actualmente (saltos y llamadas a subprogramas).

• Siempre dependen de la frecuencia del oscilador de que disponga el microcontrolador: 1 ciclo de instrucción = 4 ciclos de oscilador



Cálculo del número de ciclos máquina

• Para conseguir alcanzar una temporización dada, a partir del oscilador disponible, se puede obtener el número de ciclos máquina necesarios

$$Ciclos\_máquina = \frac{Temporización}{4Tosc}$$

$$Ciclos\_máquina = \frac{Temporización \circ fosc}{4}$$

Como ejemplo: Temporización de 15ms

Oscilador de frecuencia 4MHz

$$Ciclos\_máquina = \frac{15ms \circ 4MHz}{4} = 15000$$

Temporizaciones mediante bucles

• Para conseguir un número alto de ciclos máquina ocupando lo menos posible en la memoria de programa (menor número de instrucciones) se deberían de utilizar sucesivas repeticiones de una misma zona de código (bucles)

Bucle simple

```
CUENTA  EQU   0x20
        ...
        movlw   N
        movwf   CUENTA
BUCLE   decfsz  CUENTA
        goto    BUCLE
```

Valor de N a definir

Aportación de cada instrucción:

- movlw N: 1 ciclo máquina
- movwf CUENTA: 1 ciclo máquina
- decfsz CUENTA: N-1 ciclos si no salta + 2 ciclos al saltar
- goto BUCLE: (N-1)*2 ciclos máquina

Duración = 1 + 1 + (N-1) + 2 + (N-1) x 2 = 3N + 1 (ciclos)

En el caso anterior, si N = 0 se obtendría el valor máximo de duración del anterior fragmento de código (con el primer decremento N-1 = 255) y éste sería :

Duración máxima = 3 x 256 + 1 = 769 ciclos máquina

se podría aumentar el factor de N con instrucciones que introduzcan ciclos en el bucle

```
CUENTA  EQU   0x20
        ...
        movlw   N
        movwf   CUENTA
BUCLE   nop
        decfsz  CUENTA
        goto    BUCLE
```

Aportación por instrucción:

- movlw N: 1 ciclo máquina
- movwf CUENTA: 1 ciclo máquina
- nop: N ciclos — Nueva instrucción en el bucle
- decfsz CUENTA: N-1 ciclos si no salta + 2 ciclos al saltar
- goto BUCLE: (N-1)*2 ciclos máquina

Duración = 1 + 1 + N + (N-1) + 2 + (N-1) x 2 = 4N + 1 (c.m.)

Bucles Anidados

- Decrementando un registro en un bucle simple, se tiene la limitación en el **tamaño máximo** del mismo (8 bits -> 256 valores)
- Se pueden colocar varios **bucles seguidos**, uno a continuación de otro, en cuyo caso se irían **sumando las temporizaciones** (efecto acumulativo)
- Pero para la generación de retardos de mayor duración, se deben **anidar bucles** (un bucle dentro de otro) para conseguir un **efecto multiplicativo** en el conjunto
- Se disponen **varios registros de 8 bits** trabajando como contadores en cuenta descendente que **permiten multiplicar entre sí los retardos** producidos por cada bucle
- El **número de bucles** y **contadores** necesarios dependerá del oscilador disponible y del valor de la temporización total a realizar.
- Analicemos las posibilidades que dan 3 bucles anidados a continuación…

Tres bucles anidados:

Número de ciclos máquina:

```
        movlw   p          1
        movwf   CUENTA1    1
BUCLE1  movlw   m          1
        movwf   CUENTA2    1
BUCLE2  movlw   n          1
        movwf   CUENTA3    1
BUCLE3  decfsz  CUENTA3    1·(n-1)+2
        goto    BUCLE3     2·(n-1)        )xm
        decfsz  CUENTA2    1·(m-1)+2      )xp
        goto    BUCLE2     2·(m-1)
        decfsz  CUENTA1    1·(p-1)+2
        goto    BUCLE1     2·(p-1)
```

p, m y n se deben definir con unos valores constantes o variables mediante directivas

CUENTA1 CUENTA2 y CUENTA3 son registros auxiliares

Ciclos de máquina = (((3·(n-1)+2)+2)·m+2+2+3·(m-1))·p+3·(p-1)+2+1+1=**3nmp+4mp+4p+1**

Tres bucles anidados (II):

Ciclos de máquina = **3nmp+4mp+4p+1**

- El **valor máximo** se obtiene cuando se carga **cero** en los tres contadores en cuyo caso se deben interpretar n=m=p=256 ya que DECFSZ primero decrementa y luego evalúa si se alcanzó el cero. En ese caso:

Máximo de Ciclos máquina = 50.594.817

- Conocido el número de ciclos máquina necesarios para una temporización dada, se dispone de **dos grados de libertad** para elegir los valores de m, n y p, teniendo en cuenta que **los 3 valores deben ser enteros iguales o inferiores a 256**

Ejemplo: Se necesitan 1.000.000 ciclos de máquina para una temporización dada

Se eligen p = 15 y m = 200 y se despeja el valor de **n** necesario para alcanzar ese número de ciclos de máquina, aproximando al entero más próximo

$$n = \frac{ciclos - 1 - 4p - 4mp}{mp} = 109{,}771$$

- si n=110 → Temporización algo mayor: 1.002.061 ciclos
- si n=109 → Temporización algo menor: 993.061 ciclos

Temporizaciones mediante bucles anidados

- En el ejemplo anterior, se podría buscar la terna de valores (m,n y p) que proporciona una mayor precisión a la temporización si fuera importante en el programa

- No siempre se podría asegurar una total exactitud. Para mayor precisión se deberían de utilizar las temporizaciones hardware mediante los módulos de temporización presentes en el microcontrolador que se esté utilizando

- Si el valor necesario de ciclos de máquina es mayor que el máximo posible alcanzable con un número dado de bucles anidados, se puede incrementar el factor que multiplica a los contadores mediante la inserción de instrucciones del tipo NOP o bien incrementar el número de bucles anidados

- Cuanto más "interna" sea la inserción de estas instrucciones (en el bucle más interno de todos), mayor será el factor de multiplicación al incrementar el valor que aparece multiplicando a todos los contadores

(véase el ejemplo que aparece a continuación)


```
        movlw   p          1
        movwf   CUENTA1    1
BUCLE1  movlw   m          1
        movwf   CUENTA2    1
BUCLE2  movlw   n          1
        movwf   CUENTA3    1
BUCLE3  nop                n
        decfsz  CUENTA3    1·(n-1)+2
        goto    BUCLE3     2 ·(n-1)
        decfsz  CUENTA2    1·(m-1)+2
        goto    BUCLE2     2 ·(m-1)
        decfsz  CUENTA1    1·(p-1)+2
        goto    BUCLE1     2 ·(p-1)
```

)xm

)xp

Inserción de la instrucción NOP en el bucle más interno

El factor pasa de 3 a 4

Ciclos de máquina = (((3·(n-1)+2+n)+2)·m+2+2+3·(m-1))·p+3·(p-1)+2+1+1=4nmp+4mp+4p+1

Temporizaciones con el simulador de MPLAB (MPLAB SIM)

• MUY IMPORTANTE: recordemos que los simuladores software **no trabajan en tiempo real**. El tiempo que tardan en simular la ejecución de una instrucción **SERÁ MUY SUPERIOR** al tiempo de ejecución de la misma (dependerá de las tareas en ejecución en el PC, del microprocesador del PC y de la frecuencia del mismo).

• En las simulaciones, las **temporizaciones pueden resultar muy tediosas** porque el tiempo de simulación y permanencia en esos bucles resulta enormemente largo (sobre todo si se pretende usar la opción ***"Animate"*** para visualizar la simulación . Para salvar este inconveniente, resulta aconsejable **inhabilitar estos retardos** o bien **reducir el valor a cargar en los contadores durante la fase de simulación**.

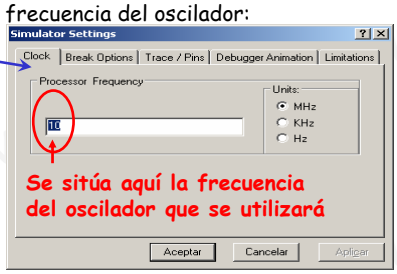
• Una vez comprobado el **funcionamiento correcto** del programa con el simulador, hay que **acordarse de volver a habilitar las temporizaciones** o **restituir los valores** válidos **de los contadores** (depende de lo que se haya hecho) de cara a su ejecución en tiempo real durante la fase de emulación, depuración (p.e. con el MPLAB-ICD2) y la posterior grabación del microcontrolador.

• En cualquier caso, el MPLAB SIM dispone de una **utilidad que permite valorar el tiempo** que tardaría en ejecutarse una determinada zona de código: ***Stopwatch***

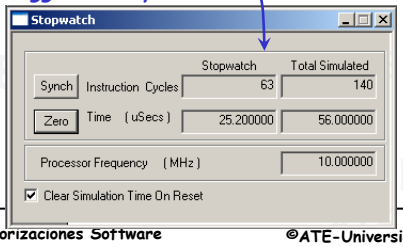
Cómo medir tiempos con MPLAB SIM

• La utilidad ***Stopwatch*** realmente es un **contador de ciclos de máquina** simulados y que a partir de la frecuencia especificada para el oscilador que se utilizará en la aplicación real, nos determina el tiempo de ejecución del intervalo simulado (de manera continua, con animación, paso a paso, etc.)

• Por lo tanto será necesario primero fijar la frecuencia del oscilador:
Debugger > Settings > Clock



• Posteriormente, y de cara a la simulación, se podría visualizar ya la ventana del cronómetro:
Debugger > Stopwatch



Synch: sincroniza contador y total de sim.

Zero: pone a cero Stopwatch