Informe 12
PIC Contador

Laboratorio de Arquitectura del Computador

Elaborado por:
Tomás Guzmán, 21615008

1 Marco teórico

En esta experiencia vamos a reunir todo lo que hemos construido en las experiencias anteriores con el PIC.

En este caso, vamos a recalcar que vamos a usar para poder implementar el PIC contador:

- Inicialización de prescalar en OPTION_REG, usaremos 256 en este caso, o bien 111 en PS<2:0>
- Inicialización del INTCON con interrupciones globales e interrupción de TMR0.
- Carga de 217 a TMR0 para obtener overflows cada 10 ms.
- Carga de 100 a un registro de uso general, COUNT, para hacer 100 ciclos de 10 ms y lograr 1 segundo.

2 Implementación

```
    LIST P=16F84

OPT     EQU 01H
TMR0    EQU 01H
PCL     EQU 02H
STATUS  EQU 03H
PORTA   EQU 05H
PORTB   EQU 06H
TRISA   EQU 05H
TRISB   EQU 06H
INTCON  EQU 0BH

SEGUNI  EQU 11H
SEGDEC  EQU 12H
MINUNI  EQU 13H
MINDEC  EQU 14H
```

```
COUNT   EQU 10H
COUNT1  EQU 15H

#DEFINE T0IF    INTCON, 2
#DEFINE RA4     PORTA, 4
#DEFINE Z       STATUS, 2

#DEFINE BANK0   BCF STATUS, 5
#DEFINE BANK1   BSF STATUS, 5

    ORG 0
    GOTO INICIO

INICIO ORG 10
    BANK1
    MOVLW   B'11111111'
    MOVWF   TRISA       ; PORTA es de entrada ahora
    CLRF    TRISB

    MOVLW   B'00000111' ; Prescalar de 256
    MOVWF   OPT         ; Se mueve el prescalar de 256 a OPTION_REG
    BANK0

    CLRF    SEGUNI
    CLRF    SEGDEC
    CLRF    MINUNI
    CLRF    MINDEC      ; Borradas variables que llevan tiempo

    MOVLW   B'10100000'
    MOVWF   INTCON      ; Activación de interrupciones globales y de TMR0


    MOVLW   D'217'
    MOVWF   TMR0        ; Cargando 217 a TMR0 para obtener 10 ms
    MOVLW   D'100'
    MOVWF   COUNT       ; Cargado 100 en COUNT, para obtener 1000 ms

MAIN
    BTFSS   RA4
    GOTO    FIN         ; Si el boton esta presionado se va a FIN

CARGA
```

```
    MOVF    SEGUNI, 0   ; Se carga SEGUNI a W
    CALL    CONVERT
    MOVWF   PORTB       ; Se mueve el valor que dio CONVERT a PORTB
    GOTO    MAIN

CONVERT
    ADDWF   PCL, 1          ; Se suma W a PCL y se guarda en PCL
    RETLW   B'11000000'     ; Se guarda en W el código de segmentos de 0
    RETLW   B'11111001'     ; Se guarda en W el código de segmentos de 1
    RETLW   B'10100100'     ; Se guarda en W el código de segmentos de 2
    RETLW   B'10110000'     ; Se guarda en W el código de segmentos de 3
    RETLW   B'10011001'     ; Se guarda en W el código de segmentos de 4
    RETLW   B'10010010'     ; Se guarda en W el código de segmentos de 5
    RETLW   B'10000010'     ; Se guarda en W el código de segmentos de 6
    RETLW   B'11111000'     ; Se guarda en W el código de segmentos de 7
    RETLW   B'10000000'     ; Se guarda en W el código de segmentos de 8
    RETLW   B'10011000'     ; Se guarda en W el código de segmentos de 9

    ORG 4
    GOTO INTER

INTER ORG 90
    DECFSZ  COUNT, 1        ; Se disminuye COUNT en 1 y se guarda en COUNT
    GOTO    RESET_TMR0      ; Brincar si no ha pasado un segundo (COUNT <>
 0)
    GOTO    SEG             ; Brincar si paso un segundo (COUNT = 0)

RESET_TMR0
    MOVLW   D'217'
    MOVWF   TMR0            ; Se reestablece TMR0 a 217 (10 ms)
    BCF     T0IF            ; Se apaga la flag de interrupcion de TMR0 en
INTCON
    RETFIE                  ; Se sale de interrupcion

SEG
    INCF    SEGUNI, 1       ; Se incrementa unidades de segundo, se guarda
 en SEGUNI
    MOVF    SEGUNI, 0       ; Se mueve SEGUNI a W
    SUBLW   D'10'           ; Se resta 10 a W
    BTFSS   Z               ; Si el resultado anterior es cero, Z = 1
    GOTO    RESET_COUNT
    CLRF    SEGUNI          ; Se limpia SEGUNI
```

```
RESET_COUNT
    MOVLW   D'100'
    MOVWF   COUNT           ; Se reestablece COUNT a 100
    GOTO    RESET_TMR0

FIN
    CLRF    INTCON          ; Se desactivan todas las interrupciones
    GOTO    CARGA

END
```

Este código es sencillo, conociendo el funcionamiento del PIC a esta altura.

Básicamente cuenta con un ciclo de espera, como en la experiencia anterior, en donde se mantiene cargando valores a PORTB.

Note que en MAIN si se pulsa un botón se salta a FIN donde se desactivan las interrupciones (haciendo que INTCON sea 0).

En CARGA que es la parte del ciclo que más trabaja, se llama constantemente a CONVERT.

Para modificar los números mostrados en el display esta rutina se vale del valor cargado a W, y brinca i líneas $i \in [0,9]$. W es sumado a PCL para lograr este valor.

Una vez elegida la línea se va de regreso a MAIN para volver a hacer el ciclo.

W cambiará, en este caso, cada segundo, dado que TMR0 fue configurado para hacerlo de esta forma.

Cuando COUNT = 0, entonces ya se cumplió un segundo (es decir, TMR0 ya se desbordó 100 veces, tomándole cada desborde 10 ms). Luego, si esto sucede, se aumenta el valor del SEGUNI en 1 y el mismo es cargado en W para ser usado por las rutinas del ciclo MAIN/CARGA.

En un momento SEGUNI llegará a 10, valor que no debería ser mostrado por el contador, dado 10 decimal es una A en hexadecimal y eso no es legible en el mundo humano (por lo menos no para el uso general).

Como no se desea esto se aprovecha que se cargó SEGUNI a W, se le resta 10 y si el resultado es 0, entonces se hace un CLRF sobre SEGUNI, es decir, se vuelve a 0. Esto permite hacer el contador nunca llegue a 10, sino que se mantenga entre 0 y 9.

Para determinar si esta resta $W - 10 = 0$ se hace uso de

```
    SUBLW   D'10'           ; Se resta 10 a W
    BTFSS   Z               ; Si el resultado anterior es cero, Z = 1
```

En donde Z es el segundo bit de status, que se prende si la última operación es igual a cero.

2.1 COMPILACIÓN

Este programa en extensión .asm fue compilado haciendo uso de MPASMWIN.

Este programa se incluyó en el PATH de Windows, por lo cual se puede ejecutar desde cualquier terminal.

Su sintaxis es sencilla, en este caso:

$ mpasmwin practica12.asm

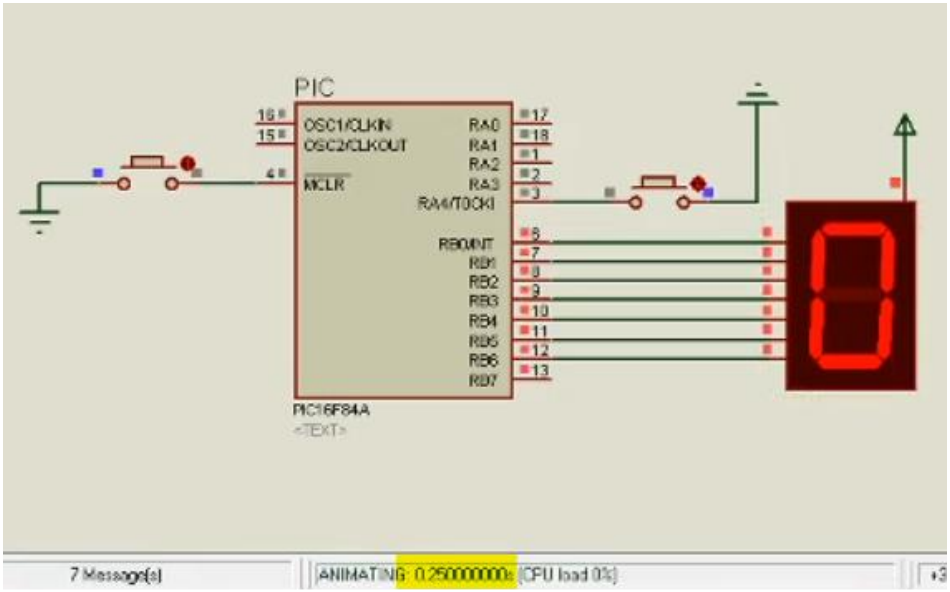
Produce varios archivos, entre ellos el practica12.hex. Este último se usará en el PIC implementado en Multisim para visualizar la funcionalidad.

3 SIMULACIÓN

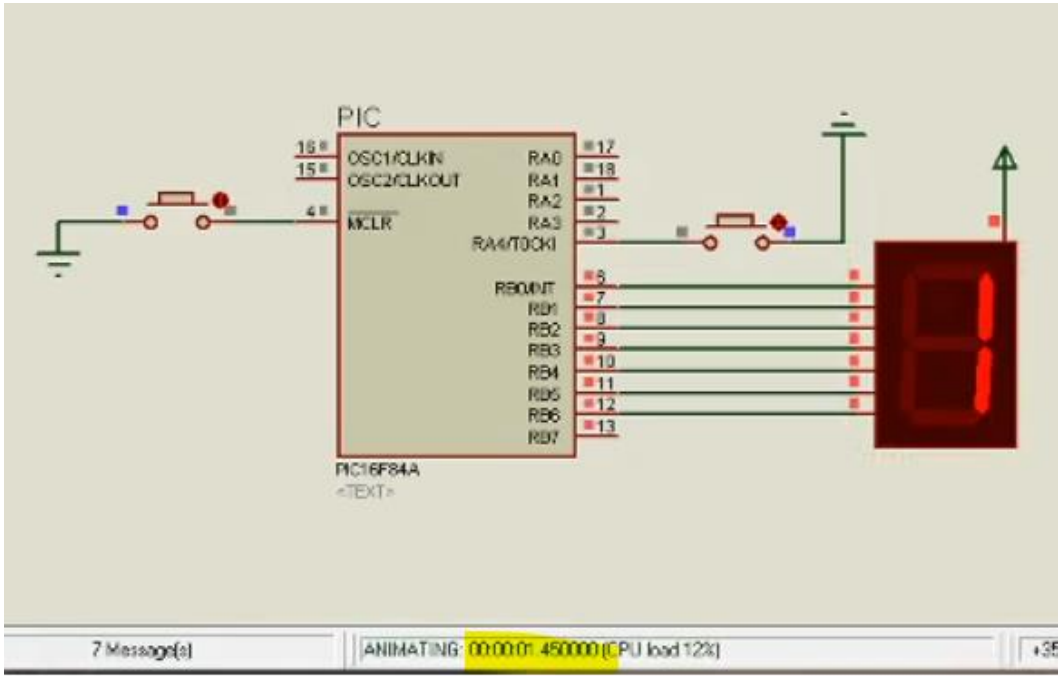
Se tomó video en Windows para poder tener exactitud sobre lo que se debe mostrar. Se realizó esta simulación en Proteus.

En primera instancia, el contador comienza en 0.

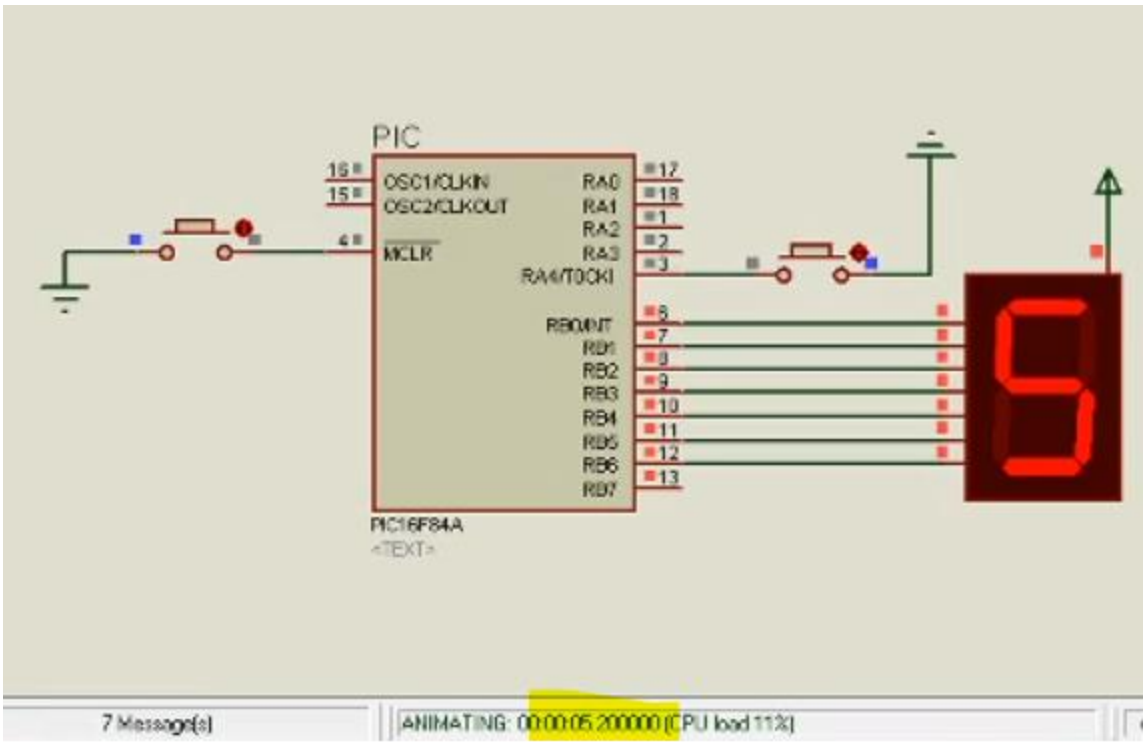
Esto se ve así en el intervalo de tiempo $t \in [0, 1)$. La próxima imagen es de $t = 0.25\ s$

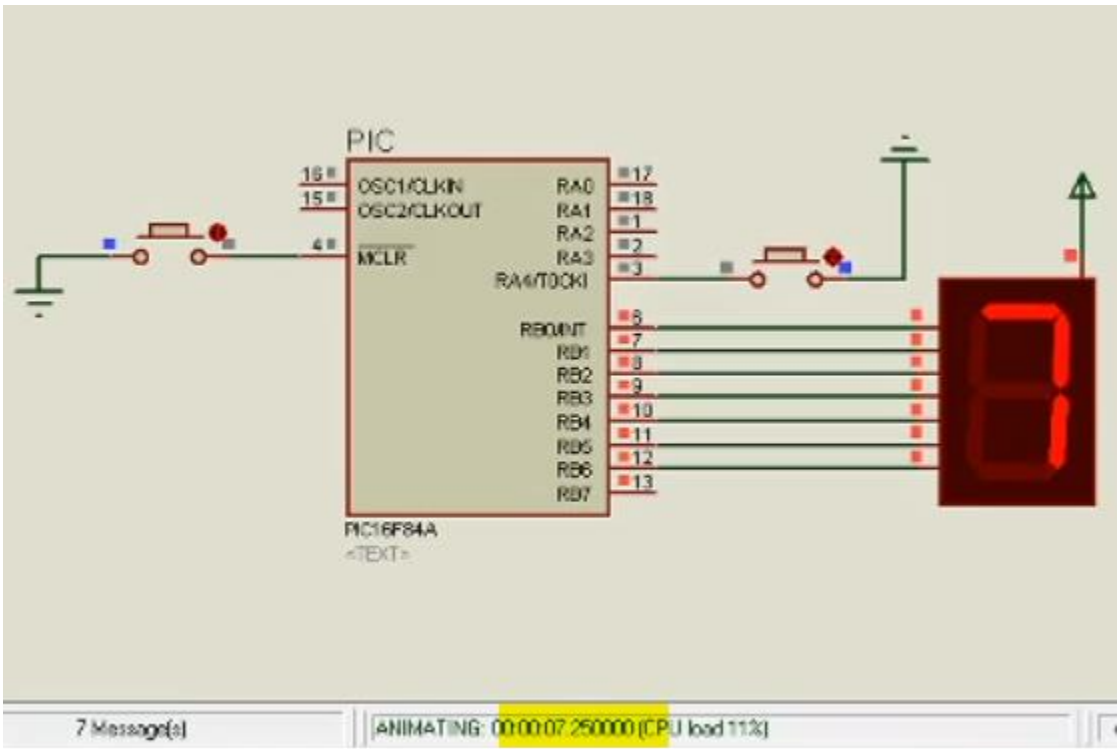


Ahora veamos para $t \in [1, 2)$

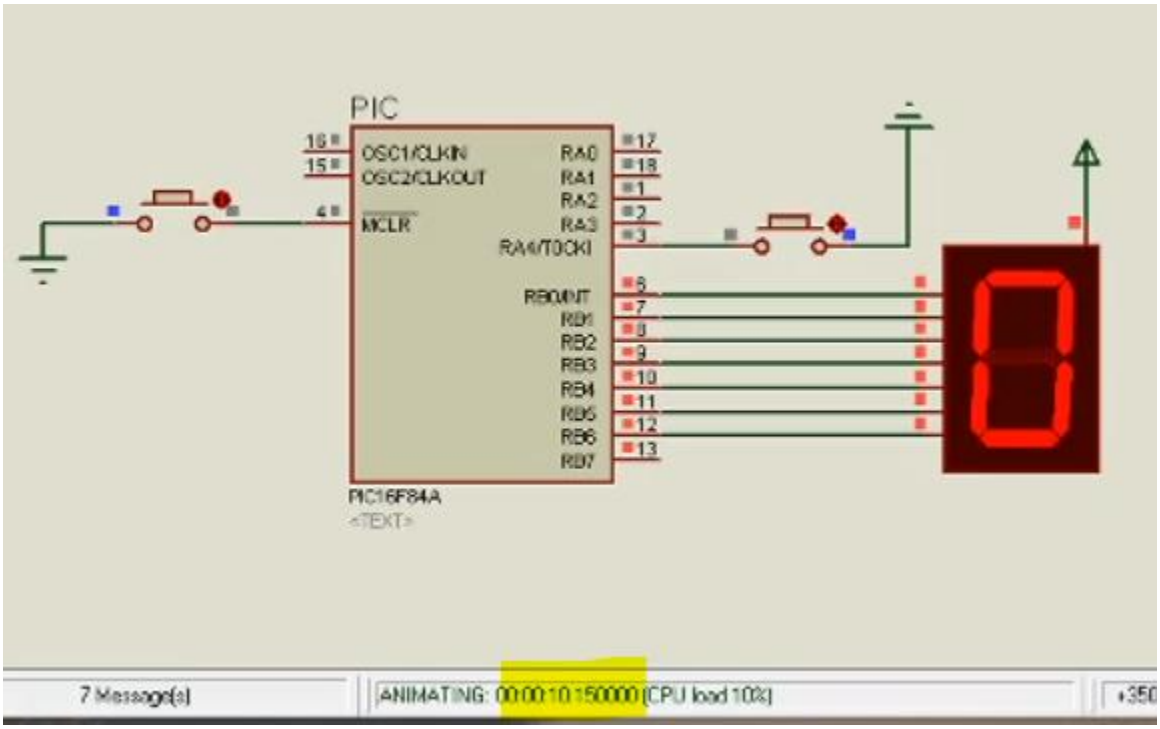


Esto ocurre para todos los números.


PIC
16
15
4
OSC1/CLKIN
OSC2/CLKOUT
MCLR
RA0
RA1
RA2
RA3
RA4/T0CKI
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
17
18
1
2
3
6
7
8
9
10
11
12
13
PIC16F84A
<TEXT>
7 Message(s)
ANIMATING: 00:00:05.200000 (CPU load 11%)


PIC
16
15
4
OSC1/CLKIN
OSC2/CLKOUT
MCLR
RA0
RA1
RA2
RA3
RA4/T0CKI
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
17
18
1
2
3
6
7
8
9
10
11
12
13
PIC16F84A
<TEXT>
7 Message(s)
ANIMATING: 00:00:07.250000 (CPU load 11%)

Ahora otro caso que nos interesa es el de $t > 10$. En este caso sabemos que ya este número (10) no se va a mostrar en pantalla, y que SEGUNI será devuelto a 0.



Como se puede apreciar, en $t = 10.15\ s$ el contador es nuevamente 0. Si se llevaran decenas en este algoritmo, en este momento se hace que SEGUNI sea cero y SEGDEC se le suma uno.

4 CONCLUSIONES

- Los PICs son pequeños dispositivos con una gran capacidad
    - Pueden usar sus pines de entrada y salida
    - Pueden llevar contadores
- Los PICs poseen una gran variedad de registros de uso especial para poder llevar una gran cantidad de operaciones.
- Haciendo uso de esta multitud de registros, la imaginación es el límite. Este pequeño reloj digital de unidades es apenas una de las implementaciones. Un PIC también se puede unir con una pantalla LCD para crear aún más símbolos y mostrarle información al usuario.