Primera Práctica

Diseño de Sistemas Basados en Microprocesador

David Morales Sáez
Alejandro Sánchez Medina

Índice

Objetivos	2
Fundamento Teórico	2
Algoritmo	4
Código	5
Bibliografía	10

Objetivos

Se diseñará un programa que realice las siguientes acciones sobre la placa PicSchool.

1º Sobre el display 7 segmentos se mostrarán los números del 0 hasta el 9 consecutivamente. Cuando se alcance el máximo se procederá a retroceder hasta el cero, pero ahora con el punto iluminado. Alcanzado el cero se repetirá el ciclo. Los cambios deben producirse a intervalos de segundo. El intervalo debe obtenerse con un bucle anidado calculando el número de iteraciones necesarias dada la frecuencia de reloj utilizada.

2º Se introducirá el tratamiento de dos pulsadores. Un pulsador permitirá congelar el display. Esto significa que cuando se pulse, el número que aparece en el display en ese momento quedará congelado, pero la cuenta interna continuará. Cuando vuelva a pulsarse el mismo botón continuará mostrándose la cuenta actualizándose el display al valor interno adecuado. El segundo pulsador debe resetear la cuenta interna a cero. Y a partir de ahí empezará a incrementarse.

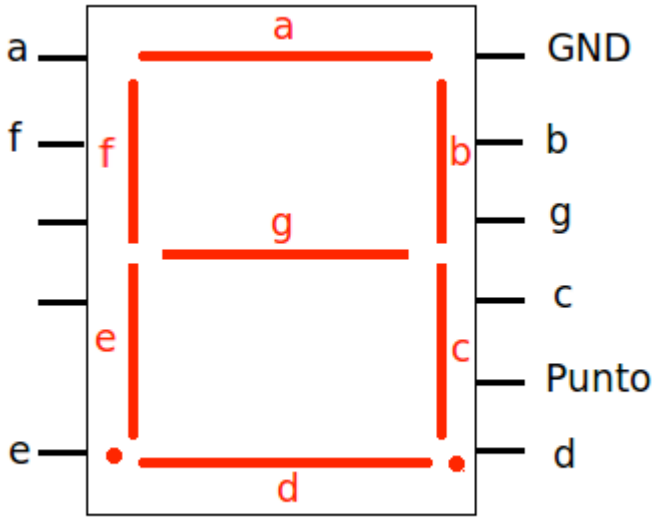
3º Se programará el timer interno del procesador para que se dispare una interrupción cada 5 segundos. Cuando se dispare la interrupción el display parpadeará con el número que se esté representando en ese momento. Parpadeará tres veces a intervalos de una décima de segundo. Al retornar de la interrupción se continuará la cuenta.

Fundamento Teórico

El PIC (Periphera Interface Controller) es un microcontrolador creado por Arizona Microchip Technology cuya arquitectura se fundamenta en el modelo Harvard, tiene una memoria encauzada de dos etapas de ejecución y la arquitectura del repertorio se basa en el modelo RISC.

Nosotros trabajamos con el PIC16F84, un PIC con una arquitectura de 8 bits y 18 pines. En la arquitectura básica, se utiliza el registro W como un acumulador genérico, mientras que la RAM es vista como un archivo de registros. Permite un anidamiento de llamadas a rutinas en la pila, pero no para datos, y dispone de una EEPROM regrabable.

Para realizar la práctica, hemos utilizado un display de siete segmentos, con la siguiente configuración:



Display 7 segmentos

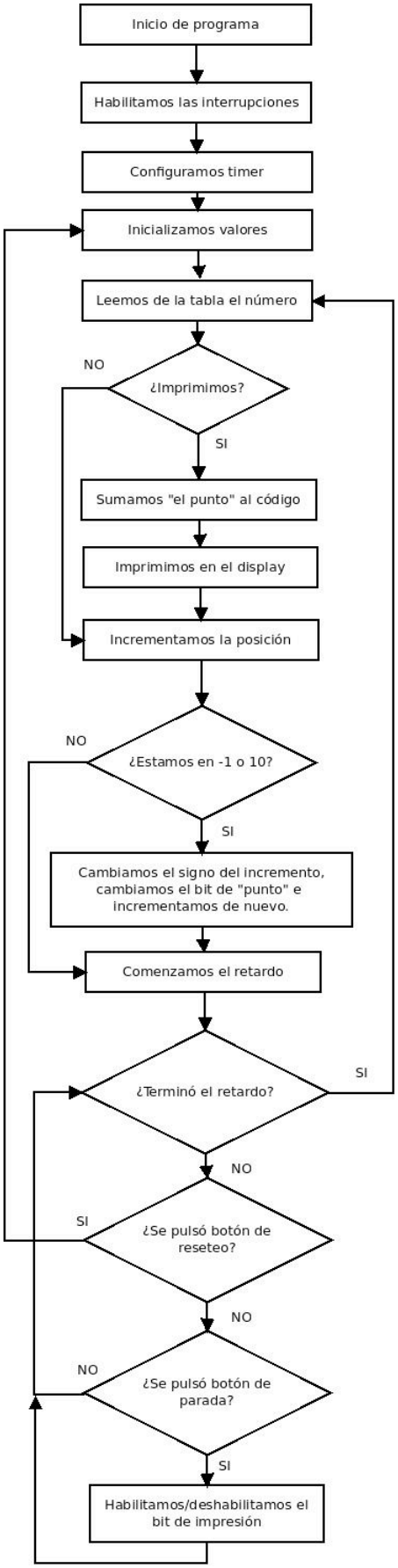
Además, hemos llevado las salidas del puerto B al display de la siguiente forma:

Nº	a	b	c	d	e	f	g	.
0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	0	0
2	1	1	0	1	1	0	1	0
3	1	1	1	1	0	0	1	0
4	0	1	1	0	0	1	1	0
5	1	0	1	1	0	1	1	0
6	1	0	1	1	1	1	1	0
7	1	1	1	0	0	0	0	0
8	1	1	1	1	1	1	1	0
9	1	1	1	1	0	1	1	0

Códigos display

Para conocer el retardo que hay que aplicar, tuvimos que hallar la tiempo de ciclo, que es la inversa de la frecuencia de reloj (1 microsegundo). Con esto, vemos que el CPI es 10^6 , por lo que, con una mera regla de tres ($T = N_{inst}*CPI*T_{ciclo}$ -> $N_{inst} = T / (CPI*T_{ciclo})$ -> $N_{inst} = 1/(10^6*10^{-12}) = 10^6$) hallamos el número de instrucciones que debíamos usar para llegar a la pausa de un segundo entre dos números y, tras ver las instrucciones que tenemos, ponemos el retardo en función a este. Cabe decir, que esto no es totalmente exacto, ya que no podemos asegurar que haga exactamente un segundo (hacemos 999977 instrucciones en lugar de un millón), además de tener las interrupciones por las pulsaciones.

Algoritmo


Inicio de programa
Habilitamos las interrupciones
Configuramos timer
Inicializamos valores
Leemos de la tabla el número
¿Imprimimos?
NO
SI
Sumamos "el punto" al código
Imprimimos en el display
Incrementamos la posición
¿Estamos en -1 o 10?
NO
SI
Cambiamos el signo del incremento, cambiamos el bit de "punto" e incrementamos de nuevo.
Comenzamos el retardo
¿Terminó el retardo?
SI
NO
¿Se pulsó botón de reseteo?
SI
NO
¿Se pulsó botón de parada?
NO
SI
Habilitamos/deshabilitamos el bit de impresión

Código Fuente

```
                List    p=16f84
                include "P16F84.INC"

__CONFIG _WDT_OFF & _RC_OSC

Prm             equ     0x0d
Tmp             equ     0x0e
Cont            equ     0x0f
Vuelta          equ     0x10
Viejo           equ     0x11
Print           equ     0x12
Incr            equ     0x13
Ctrl            equ     0x14
Rts             equ     0x15
Wold            equ     0x16
STold           equ     0x17
CTimer          equ     0x18
BTimer          equ     0x19
NTimer          equ     0x1a
BTimer2         equ     0x1b
BTimer3         equ     0x1c

                org     0x00
                goto    Inicio

                org             0x04

                                                ; Salvamos el contexto
                movwf   Wold
                swapf   STATUS,w
                movwf   STold

                                                ; Rutina de timer
                movlw   1
                addwf   CTimer,1
                movlw   40
                subwf   CTimer,0
                btfss   STATUS, Z
                goto    Retu
                movlw   0
                movwf   CTimer

                                                ; Parpadeos
                movlw   3
                movwf   BTimer

BucleP:                                         ; "Apagamos" el Display
                movf    PORTB, 0
                movwf   NTimer
                movlw   b'00000000'
                movwf   PORTB

                                                ; Retardo interno
                movlw   0fah
                movwf   BTimer2

BucleP2:        movlw   40
                movwf   BTimer3

BucleP3:        decfsz  BTimer3,1
```

```
                goto    BucleP3
                decfsz  BTimer2,1
                goto    BucleP2

                                        ; Fin Retardo
                                        ; "Encendemos" el Display
                movf    NTimer,0
                movwf   PORTB
                movlw   0fah
                movwf   BTimer2

BucleP4:        movlw   40
                movwf   BTimer3

BucleP5:        decfsz  BTimer3,1
                goto    BucleP5
                decfsz  BTimer2,1
                goto    BucleP4
                decfsz  BTimer,1
                goto    BucleP
                                        ; Fin parpadeos
                                        ; Fin de rutina

                                        ; Restituimos el contexto
Retu:           bcf     INTCON, 2
                swapf   STold, w
                movwf   STATUS
                swapf   Wold, f
                swapf   Wold, w
                retflw

                                        ; Nos creamos una tabla
Tabla           addwf   PCL, F
                retlw   b'11111100'
                retlw   b'01100000'
                retlw   b'11011010'
                retlw   b'11110010'
                retlw   b'01100110'
                retlw   b'10110110'
                retlw   b'10111110'
                retlw   b'11100000'
                retlw   b'11111110'
                retlw   b'11110110'
                                        ; Comprueba si hay un cambio en la pulsación
Comp            btfsc   Viejo, 0
                goto    Sigu
                bsf     Ctrl, 0
                bsf     Viejo, 0
                goto    Sigu

Comp2   btfsc   Viejo, 1
                goto    Sigu2
                btfsc   Ctrl, 1
                goto    c2
                bsf     Ctrl, 1
                goto    c3

c2              bcf     Ctrl, 1

c3              bsf     Viejo, 1
                goto    Sigu2

Retardo movlw   05h
                movwf   Rts
```

```
                nop
                nop
                nop

Retard2 decfsz  Rts, 1
                goto    Retard
                goto    final

Retard          movlw   0ffh
                movwf   Tmp

                                        ; Comprobación del pulsador de reseteo
                btfsc   PORTA, 0
                goto    Comp            ; Comprobamos el bit viejo
                bcf     Viejo, 0

                                        ; Comprobación del pulsador de pausa
Sigu            btfsc   PORTA, 1
                goto    Comp2           ; Comprobamos el bit viejo
                bcf     Viejo, 1

Sigu2           decfsz  Prm, 1
                goto    continue
                goto    Retard2

continua decfsz Tmp, 1
                goto    continua
                goto    Retard

final           return

Inicio
                movlw   0
                movwf   CTimer
                                        ; Habilitamos las interrupciones a nivel GLOBAL
                bsf     INTCON, 7
                bsf     INTCON, 5
                                        ; Configuramos el TMR0
                bsf     STATUS, RP0     ; Vamos al banco 1
                movlw   b'00000111'     ; Seleccionamos la configuración del timer
                                ; 0 -> PSA: El preescaler se asigna a TMR
                                ; 0 -> TOSE: El incremento es por flanco ascendente
                                ; 0 -> TOCS: La entrada del reloj es por ciclo el reloj interno
                                        ; 111 -> PSX: El preescaler está a 1:25
                movwf   OPTION_REG      ; La guardamos en OPTION
                bcf     STATUS,RP0      ; Volvemos al banco 0
                bsf     STATUS, RP0
                clrf    TRISB
                bcf     STATUS, RP0
                clrf    PORTB

                                        ; Ponemos el contador a 0
                movlw   0
                movwf   Cont
                                        ; Ponemos vuelta a 0
                movwf   Vuelta
                                        ; Ponemos los flags de los pulsadores a 0
                movwf   Ctrl
                        ; Ponemos ambos bits de viejo a 1 (posición por defecto del pulsador)
                movlw   3
                movwf   Viejo
                                        ; Ponemos Incr a 1 por defecto
                movlw   1
                movwf   Incr
```

```
Bucle   movf   Cont,0
        call   Tabla                ;Cada CALL Tabla vale por 3 Ciclos
                                    ; Comprobamos si hay que imprimir
        btfsc  Ctrl, 1
        goto   Aum
                                    ; Sumamos la vuelta para añadir el punto
        addwf  Vuelta, 0
        movwf  PORTB

                                    ; Aumentamos el contador
Aum     movf   Incr,0
        addwf  Cont, 1

                                ; Comienzo de retardo con las comprobaciones de límite
        movlw  0ffh
        subwf  Cont, 0

        btfss  STATUS, Z
        goto   Sigue                ; No hemos llegado a 0 (decrementando)

               ; HASTA AQUI, 13 CICLOS IGNORANDO LA ESCRITURA EN EL PUERTO
        movlw  1
        addwf  Cont, 1
                                    ; Indicamos que aumentamos el iterador
        movlw  1
        movwf  Incr

                                    ; No mostramos el punto
        movlw  0
        movwf  Vuelta
        goto   Reta

               ; SI ES -1 TENDREMOS UNA PENALIZACION DE 3 CICLOS

Sigue   movlw  0Ah
        subwf  Cont, 0
        btfss  STATUS, Z
        goto   Reta                 ; No hemos llegado a 10

                                    ; Ponemos el valor a 9
        movlw  1
        subwf  Cont, 1
                                    ; Indicamos que disminuimos el iterador
        movlw  -1
        movwf  Incr
                                    ; Indicamos que hay que poner el punto
        movlw  1
        movwf  Vuelta

               ; SI ES 11 TENDREMOS UNA PENALIZACION DE 6 CICLOS

Reta    movlw  0ffh
        movwf  Prm
        call   Retardo

                                    ; Comprobamos si se activó el botón de reseteo
        btfss  Ctrl, 0
        goto   c1

                                    ; Se resetea la cuenta
        clrf   Cont
        clrf   Vuelta
        bcf    Ctrl, 0
```

```
                movlw   1
                movwf   Incr

                        ; SI SE RESETEA, TENEMOS UNA PENALIZACION DE 5 CICLOS

c1              goto    Bucle

                        ; EL CASO ESTANDARD, ES DE 23 CICLOS
                end
```

Bibliografía

- Diapositiva PIC16f84, dada por el profesor