Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Electrónica

Microcontroladores

Materia

Práctica No. 2

Camarillo Bautista Alfredo, González Escalona Miguel Ángel, Lázaro Bonilla Ramiro, López Arce Roberto

Integrantes del equipo

Ingeniería en mecatrónica

Carrera

Ricardo Álvarez González

Docente

22 de febrero de 2020, Primavera 2021

Fecha de entrega

Contenido

Marco teórico

El pic 18f4550 es un microcontrolador de 8 bits de la empresa Microchip. Este microcontrolador cuenta con una gran cantidad de memoria RAM, diferentes módulos de comunicación, una gran cantidad de pines de entrada y salida y algunas otras grandes cualidades. Algunas de sus características son:

- 40 pines tipo DIP
- Interfaz USB 2.0 de alta velocidad, EEPROM 256 bytes
- Memoria RAM 2048 bytes, EEPROM 256 bytes
- Memoria de programa (memoria flash) 32 kb
- Voltaje de operación 2 a 5.5 V
- Frecuencia máxima 48 MHz
- 35 pines de entrada / salida

Las instrucciones utilizadas en la práctica se enlistarán y explicarán a continuación.

Instrucción	Sintaxis	Operación	Palabras y Ciclos	Descripción
Bcf	[label] BCF f,b[,a]	0 -> f<b>	1 palabra 1 ciclo	El bit 'b' en el registro 'f' es limpiado. Si 'a' es 0, el Access bank será seleccionado, sobre escribiendo el valor de BSR. Si 'a' es 1, entonces el banco será seleccionado según el valor de BSR (default).
Bra	[label] BRA n	(PC) + 2 + 2n -> PC	1 palabra 2 ciclos	Suma el complemento a 2 '2n' al pc
Bsf	[label] BSF f,b[a]	1 -> f<b>	1 palabra 1 ciclo	El bit 'b' en el registro 'f' es colocado. Si 'a' es 0 el Access bank será seleccionado, sobre escribiendo el valor de BSR. Si 'a' es 1, entonces el banco será seleccionado según el valor de BSR (default).
Btg	[label] BTG f,b[,a]	¬(f<b>) -> f<b>	1 palabra 1 ciclo	El bit 'b' en la memoria de datos localizada en 'f' se invierte. Si 'a' es 0, el banco de acceso será seleccionado, sobre escribiendo el valor de BSR. Si 'a' = 1, entonces el banco será seleccionado según el valor de BSR (default).
Retfie	[label] RETFIE [s]	(TOS) → PC, 1 → GIE/GIEH or PEIE/GIEL, if s = 1 (WS) → W, (STATUSS) → STATUS, (BSRS) → BSR, PCLATU, PCLATH are unchanged.	1 palabra 2 ciclos	Regresa de la interrupción. El stack se llena y la cima del stack (TOS) se carga dentro de la PC. Las interrupciones son habilitadas estableciendo el bit

				de habilitación de interrupción global de prioridad alta o baja.
Btfss	[label] BTFSS f,b[,a]	Skip if (f<b>) = 1	0 palabras 0 ciclos	Si el bit 'b' en el registro 'f' es 1, entonces la siguiente instrucción es omitida. Si el bit 'b' es 1, entonces la siguiente instrucción traída durante la instrucción ejecutada actualmente.
Call	[label] CALL k[,s]	(PC) + 4 → TOS, k → PC<20:1>, if s = 1 (W) → WS, (STATUS) → STATUSS, (BSR) → BSRS	0 palabras 0 ciclos	Es un llamado de subrutina de un rango de memoria de 2 Mbytes.
Clrf	[label] CLRF f[,a]	00h -> f 1 -> Z	1 palabra 1 ciclo	Limpia el contenido de un registro especificado.
Decfsz	[label] DECFSZ f[,d[,a]]	(f) – 1 -> dest, skip if result = 0	0 palabras 0 ciclos	Decrementa un valor al registro seleccionado, salta si es 0
Movlw	[label] MOVLW k	K -> W	1 palabra 1 ciclo	Mueve el valor literal de K al working register
Movwf	[label] MOVWF f[,a]	W -> f	0 palabras 0 ciclos	Mueve el valor de W al registro seleccionado
Nop	[label] NOP	Ninguna operación	1 palabra 1 ciclo	Ocupa simplemente el tiempo de 1 palabra y 1 ciclo
Return	[label] RETURN [s]	(TOS) → PC, if s = 1 (WS) → W, (STATUSS) → STATUS, (BSRS) → BSR, PCLATU, PCLATH are unchanged	0 palabras 0 ciclos	Regresa de la subrutina
Rlncf	[label] RLNCF f[,d[,a]	(f<n>) -> dest <n+1> (f<7>) -> dest<0>	N, Z	El contenido del registro f rota un bit hacia la izquierda
Rrncf	[label] RRNCF f[,d[,a]	(f<n>) -> dest <n-1> (f<0>) -> dest<7>	N, Z	El contenido del registro f rota un bit hacia la derecha

Directivas utilizadas

<label> EQU <value> (equate): La directiva de ensamblador ***EQU*** simplemente equipara un nombre simbólico a un valor numérico.

ORG <value> (origin): La directiva ***ORIGIN*** le dice al ensamblador donde cargar instrucciones y datos dentro de la memoria.

Display de 7 segmentos

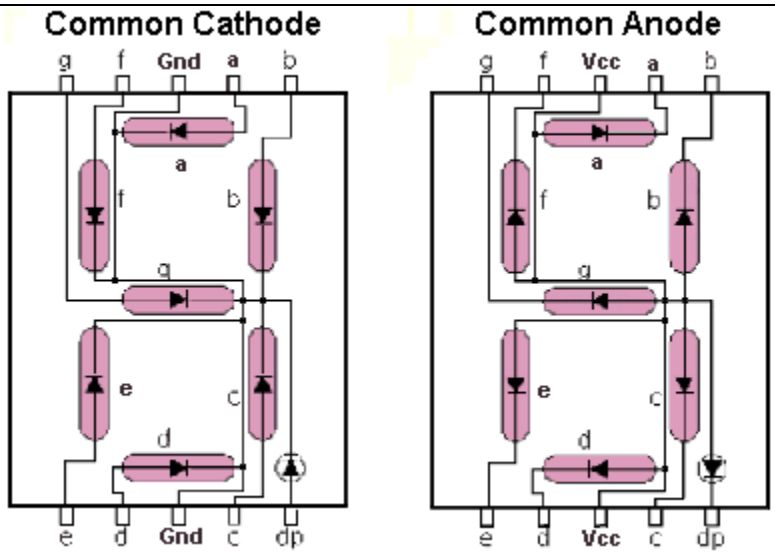

Imagen 1. Display de 7 segmentos de cátodo común (izquierda) y ánodo común (derecha).

El display de 7 segmentos es un dispositivo opto – electrónico que permite visualizar números del 0 al 9. Existen dos tipos de display, de cátodo común y de ánodo común. Especificaciones:

- Voltaje: 3 VCD
- Amperaje: 10 mA
- Número de segmentos: 7
- Cátodo común
- Color del LED: Rojo
- Posiciones de los pines con respecto al punto: Vertical
- Dimensiones: 1.8 cm x 0.9 cm x 0.4 cm

Un display de este tipo está compuesto por siete u ocho leds de diferentes formas especiales y dispuestos sobre una base de manera que puedan representarse todos los símbolos numéricos y algunas letras. Los primeros siete segmentos son los encargados de formar el símbolo y con el octavo podemos encender y apagar el punto decimal. Cada uno de los segmentos que forman la pantalla están marcados con siete primeras letras del alfabeto (‘a’ – ‘g’).

En los tipos de ánodo común, todos los ánodos de los segmentos están unidos internamente a una patilla común que debe ser conectada a potencial positivo (nivel ‘1’). El encendido de cada segmento individual se realiza aplicando potencial negativo (nivel ‘0’) por la patilla correspondiente a través de una resistencia que límite el paso de la corriente.

En los de tipo de cátodo común, todos los cátodos de los segmentos están unidos internamente a una patilla común que debe ser conectada a potencial negativo (nivel ‘0’). El encendido de cada segmento individual se realiza aplicando potencial positivo (nivel ‘1’) por la patilla correspondiente a través de una resistencia que limite el paso de la corriente.

INT0IF: INT0 External Interrupt Flag bit

1 = La interrupción externa de INT0 ocurre (debe ser 'cleared' en el software)

0 = La interrupción externa de INT0 no ocurre

Bits de control para la configuración del puerto A/D

PCFG3:PCFG0: A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Imagen 2. Bits de control para la configuración del puerto A/D.

Configuración del puerto D como salida

	7	6	5	4	3	2	1	0
TRISD	0	0	0	0	0	0	0	0
PORTD	Salida	Salida	Salida	Salida	Salida	Salida	Salida	Salida

Desarrollo práctico

Para la práctica realizada se utilizaron los siguientes materiales a enlistar.

Cantidad	Concepto
1	Pic 18F4550
1	Laptop
1	Display de 7 segmentos cátodo común
8	Resistencias de 330 Ohms
1	Software MPLAB v8.92
1	Datasheet del PIC 18F4550
1	Push button
1	Resistencia de 10k

La secuencia del contador se hizo de dos diferentes maneras, la primera realizada mediante una subrutina y la segunda mediante interrupciones.

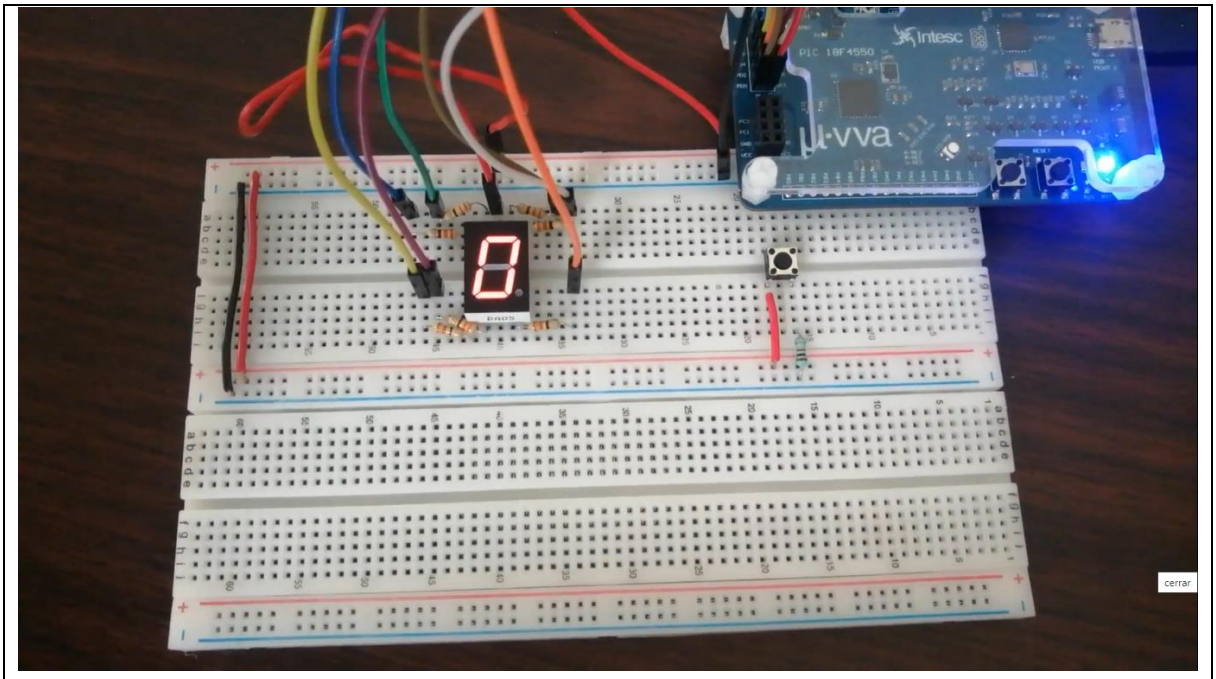

Imagen 3. Número 0_{16} mostrado en el display

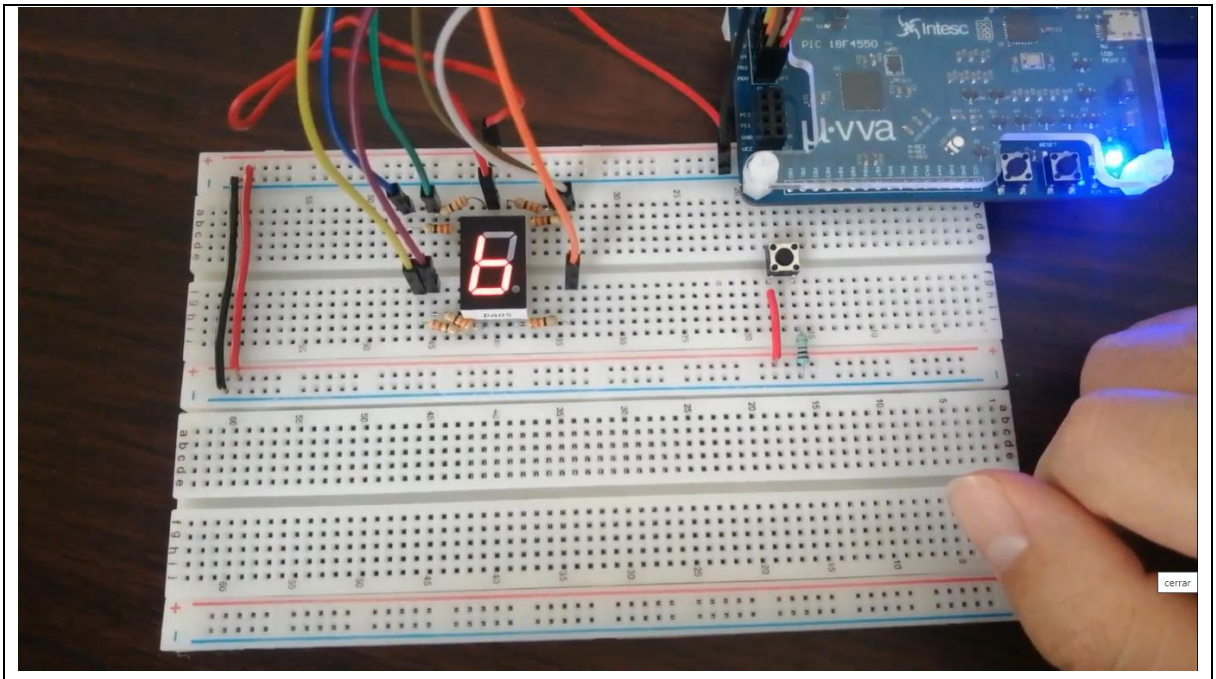

Imagen 4. Número b_{16} mostrado en el display

Código utilizando interrupciones

```
        LIST P=18F4550      ;directiva para definir el procesador
        #include <P18F4550.INC> ;definiciones de variables especificas del procesador

;*****************************************************************************
;Bits de configuración
            CONFIG FOSC = INTOSC_XT ;Oscilador interno para el uC, XT para USB
            CONFIG BOR = OFF        ;Brownout reset deshabilitado
            CONFIG PWRT = ON        ;Pwr up timer habilitado
            CONFIG WDT = OFF        ;Temporizador vigia apagado
            CONFIG MCLRE = OFF      ;Reset apagado
            CONFIG PBADEN = OFF
            CONFIG LVP = OFF
```

Imagen 5. primera parte del código

Como se puede observar en imagen x, en esa parte del código se definen las directivas para definir el procesador y variables específicas de este, además de los bits de configuración para el inicio del microcontrolador.

```
;*****************************************************************************
;Definiciones de variables
        cblock      0x0
        cont
        ciclo
        flags
        endc
;Todo lo anterior son palabras de configuración
;*****************************************************************************
;Reset vector
        ORG 0x0000
        bra     inicio
        org     0x08
        bra     RSI
```

Imagen 6. Segunda parte del código

En la segunda parte del código se definen las variables mediante la directiva ***CBLOCK***.

Sintaxis:

Cblock [expr]

Label [:increment] [,label [:increment]]

Endc

Descripción:

Define una lista de símbolos secuenciales nombrados. La lista de nombres termina cuando la directiva *endc* es encontrada.

Expr indica el valor inicial para el primer nombre en el bloque. Si no se encuentra ninguna expresión, el primer nombre recibirá un valor más grande que el último nombre en el anterior *cblock*. Si el primer *cblock* en el archivo fuente no tiene *expr*, los valores asignados empezaran con cero.

Si *increment* es especificado, entonces el siguiente *label* se le asigna el valor de *increment* más alto que el anterior *label*.

Varios nombres pueden ser escritos en una línea, separados por comas.

Cblock es útil para definir constantes en un programa y memoria de datos para la generación absoluta de código.

Después de definir las variables, se utiliza la directiva ***ORG*** explicada previamente, que indica al ensamblador donde cargar instrucciones y datos dentro de la memoria.

Sintaxis:

[label] org expr

Descripción:

Establece el origen del programa para código subsecuente en la dirección definida en *expr*. Si *label* se especifica, se le dará el valor de *expr*. Si *org* no se especifica, la generación del código empezará en la dirección cero.

Para dispositivos **PIC18**, sólo números pares en *expr* son permitidos.

```
;Inicio del programa principal

inicio  bcf     OSCCON, IRCF2, 0        ; 000, Oscilador interno a 32 KHz
        movlw   0x0F
        movwf   ADCON1, 0               ; Puertos digitales
        clrf    PORTD, 0
        clrf    TRISD, 0                ; Puerto D configurado como salida
        movlw   0x90                    ; Binario 1001 0000
        movwf   INTCON                  ; Configuramos la interrupción externa INT0
cero    movlw   0xC0                    ; Código del cero
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     f
uno     movlw   0xF9                    ; Código del uno
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     cero
dos     movlw   0xA4                    ; Código del dos
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     uno
tres    movlw   0xB0                    ; Código del tres
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     dos
cuatro  movlw   0x99                    ; Código del cuatro
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     tres
cinco   movlw   0x92                    ; Código del cinco
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     cuatro
seis    movlw   0x82                    ; Código del seis
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     cinco
siete   movlw   0xB8                    ; Código del siete
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     seis
ocho    movlw   0x80                    ; Código del ocho
        movwf   PORTD,0
        call    repite
        btfss   flags, 1, 0
        bra     siete
```

Imagen 7. Tercera parte del código.

En esta parte del código se empieza por establecer el oscilador interno a 32 KHz y configurar los puertos como digitales en el registro de control A/D (0x0F, por lo que se configuran todos los puertos como digitales).

Después, con la instrucción ***clrf*** se establece el puerto D como salida.

Después en las subrutinas se toman en cuenta los valores de salida para poder activar los display adecuados para formar los números 1, 2, …, 9 y A, B, …, F de manera que sea un contador en sistema hexadecimal. La instrucción ***BTFSS*** (bit set 'f', skip if set) verifica el bit 1 del registro flags, si es '1' entonces se saltará la siguiente instrucción, de lo contrario seguirá con la instrucción de la siguiente línea.

```
a       movlw   0x88                    ; Código del A
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     nueve
b       movlw   0x83                    ; Código del B
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     a
c       movlw   0xC6                    ; Código del C
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     b
d       movlw   0xA1                    ; Código del D
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     c
e       movlw   0x86                    ; Código del E
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     d
f       movlw   0x8E                    ; Código del F
        movwf   PORTD, 0
        call    repite
        btfss   flags, 1, 0
        bra     e
```

Imagen 8. Cuarta parte del código.

Por último, como se puede observar en la última parte del código la rutina *reparto* y *nada* sirven junto con la rutina *repite* y *llama* para lograr un retardo de aprox. 1 segundo. Primero, al llamar a la subrutina *repite*, esta mediante la instrucción ***movwf*** asigna el valor decimal '10' a la variable ciclo, después de esto sucede la subrutina *llama*, la cual llama al retardo el cuál sirve para ocupar por 128

ms el proceso del microcontrolador, para que después al terminar esta subrutina el ciclo decremente un valor a la variable *ciclo* y ejecutar de nuevo la subrutina *llama* (este ciclo termina cuando el valor de *ciclo* es 0 y se salta la instrucción siguiente, la cuál es la que llama de vuelta a la subrutina *llama*).

```
;*******RUTINA REPARTO Y NADA*********************************************************************
retardo movlw   0xff
        movwf   cont, 0
nada    nop
        decfsz  cont, 1, 0
        bra     nada
        return
;*****RUTINA REPITE Y LLAMA*************************************************************
repite  movlw   d'10'           ;La d indica que es un dato en sistema decimal
        movwf   ciclo, 0
llama   call    retardo
        decfsz  ciclo, F, 0
        bra     llama
        return                  ; Regreso de subrutina
;*****RUTINA DE SERVICIO DE INTERRUPCIÓN*************************************************
RSI     bcf     INTCON, INT0IF  ; Se apaga el bit de bandera
        btg     flags, 1        ; Bit monitor de interrupción externa
        retfie                  ; Regreso de servicio de interrupción
        END
```

Imagen 9. Quinta parte del código

En la subrutina *RSI* se hace un salto a la dirección del vector de interrupción de alta prioridad y el bit de bandera (INT0IF) del registro INTCON se apaga. La instrucción ***BTG*** cambia los valores del primer bit de *flags*, el cuál es el bit monitor de interrupción externa.

El código en el que se verifica el estado del bit 0 del puerto B sin interrupciones se muestra a continuación.

```
        LIST P=18F4550      ;directiva para definir el procesador
        #include <P18F4550.INC> ;definiciones de variables especificas del procesador

;*******************************************************************************
;Bits de configuración
            CONFIG FOSC = INTOSC_XT ;Oscilador interno para el uC, XT para USB
            CONFIG BOR = OFF        ;Brownout reset deshabilitado
            CONFIG PWRT = ON        ;Pwr up timer habilitado
            CONFIG WDT = OFF        ;Temporizador vigia apagado
            CONFIG MCLRE = OFF      ;Reset apagado
            CONFIG PBADEN = OFF
            CONFIG LVP = OFF
;*******************************************************************************
;Definiciones de variables
        cblock      0x0
        cont
        ciclo
        endc
;Todo lo anterior son palabras de configuración
;*******************************************************************************
;Reset vector
        ORG 0x0000
;Inicio del programa principal
        bcf     OSCCON, IRCF2, 0        ; 000, Oscilador interno a 32 KHz
        movlw   0x0F
        movwf   ADCON1, 0               ; Puertos digitales
        clrf    PORTD, 0
        clrf    TRISD, 0                ; Puerto D configurado como salida
cero    movlw   0xC0                    ; Código del cero
        movwf   PORTD, 0
         call    repite
         btfss   PORTB, 0, 0
         bra     f
uno     movlw   0xF9                       ; Código del uno
        movwf   PORTD, 0
         call    repite
         btfss   PORTB, 0, 0
         bra     cero
dos     movlw   0xA4                       ; Código del dos
        movwf   PORTD, 0
         call    repite
         btfss   PORTB, 0, 0
         bra     uno
tres    movlw   0xB0                       ; Código del tres
        movwf   PORTD, 0
         call    repite
         btfss   PORTB, 0, 0
         bra     dos
cuatro  movlw   0x99                       ; Código del cuatro
        movwf   PORTD, 0
         call    repite
         btfss   PORTB, 0, 0
         bra     tres
cinco   movlw   0x92                       ; Código del cinco
        movwf   PORTD, 0
         call    repite
         btfss   PORTB, 0, 0
         bra     cuatro
seis    movlw   0x82                       ; Código del seis
        movwf   PORTD, 0
```

```
          call    repite
          btfss   PORTB, 0, 0
          bra     cinco
siete     movlw   0xB8                    ; Código del siete
          movwf   PORTD, 0
          call    repite
          btfss   PORTB, 0, 0
          bra     seis
ocho      movlw   0x80                    ; Código del ocho
          movwf   PORTD,0
          call    repite
          btfss   PORTB, 0, 0
          bra     siete
nueve     movlw   0x98                    ; Código del nueve
          movwf   PORTD, 0
          call    repite
          btfss   PORTB, 0, 0
          bra     ocho
a         movlw   0x88                    ; Código del A
          movwf   PORTD, 0
          call    repite
          btfss   PORTB, 0, 0
          bra     nueve
b         movlw   0x83                    ; Código del B
          movwf   PORTD, 0
          call    repite
          btfss   PORTB, 0, 0
          bra     a
c         movlw   0xC6                    ; Código del C
          movwf   PORTD, 0
        call    repite
        btfss   PORTB, 0, 0
        bra     b
d       movlw   0xA1                  ; Código del D
        movwf   PORTD, 0
        call    repite
        btfss   PORTB, 0, 0
        bra     c
e       movlw   0x86                  ; Código del E
        movwf   PORTD, 0
        call    repite
        btfss   PORTB, 0, 0
        bra     d
f       movlw   0x8E                  ; Código del F
        movwf   PORTD, 0
        call    repite
        btfss   PORTB, 0, 0
        bra     e

;*******RUTINA REPARTO Y NADA*************************************************************
retardo movlw   0xff
        movwf   cont, 0
nada    nop
        decfsz  cont, 1, 0
        bra     nada
        return
;******RUTINA REPITE Y LLAMA***************************************************************
repite  movlw   d'10'           ;La d indica que es un dato en sistema decimal
        movwf   ciclo, 0
llama   call    retardo
         decfsz  ciclo, F, 0
         bra     llama
         return
         END
```

Imagen 10. Código 1 de la secuencia del display de 7 segmentos

Cálculos

No aplica

Simulación

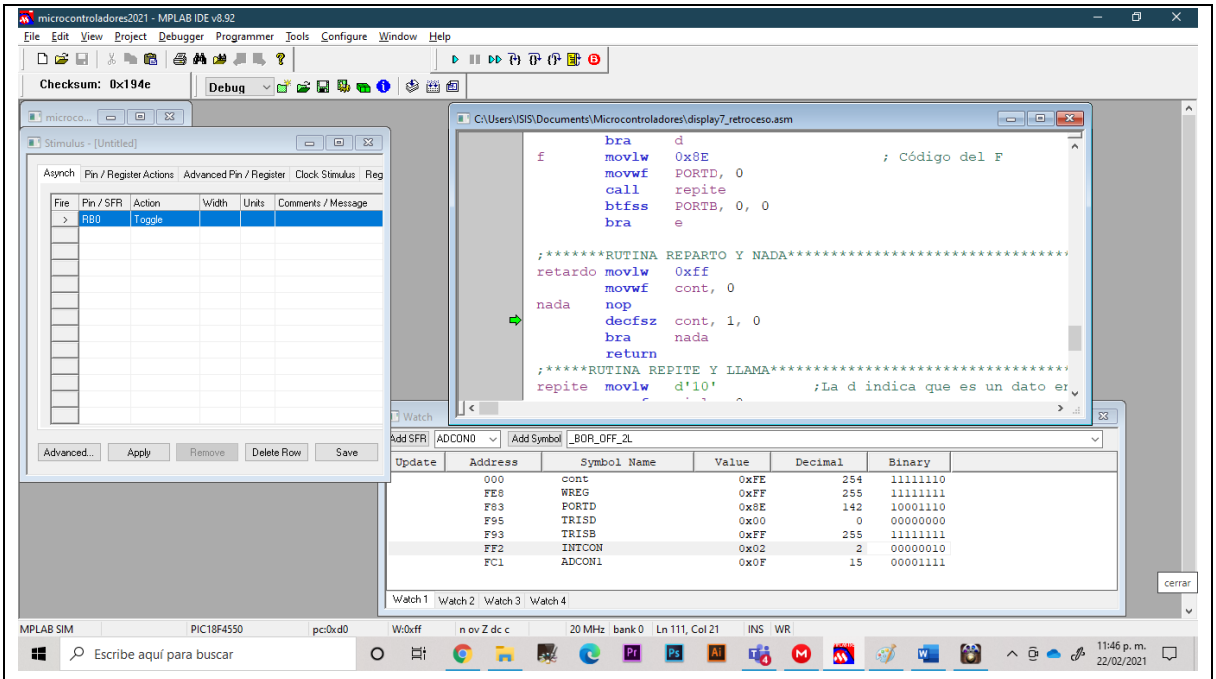

Imagen 11. Simulación del código del display de 7 segmentos con retroceso en MPLAB

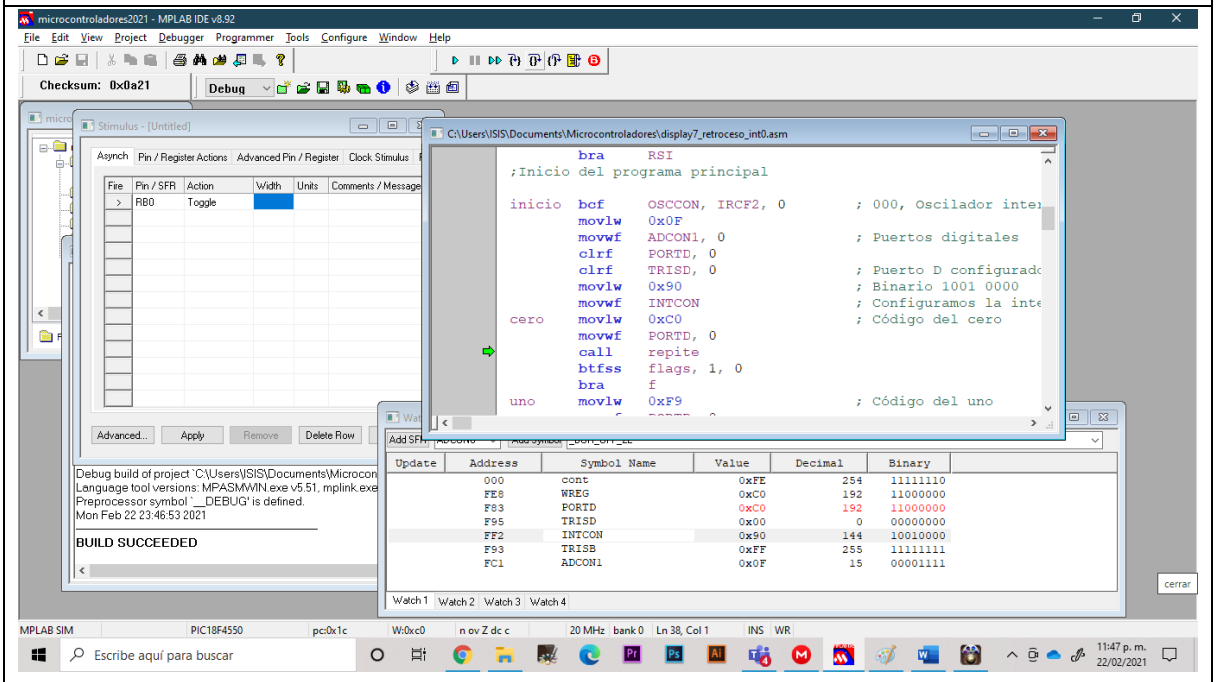

Imagen 12. Simulación del código del display de 7 segmentos con interrupciones en MPLAB

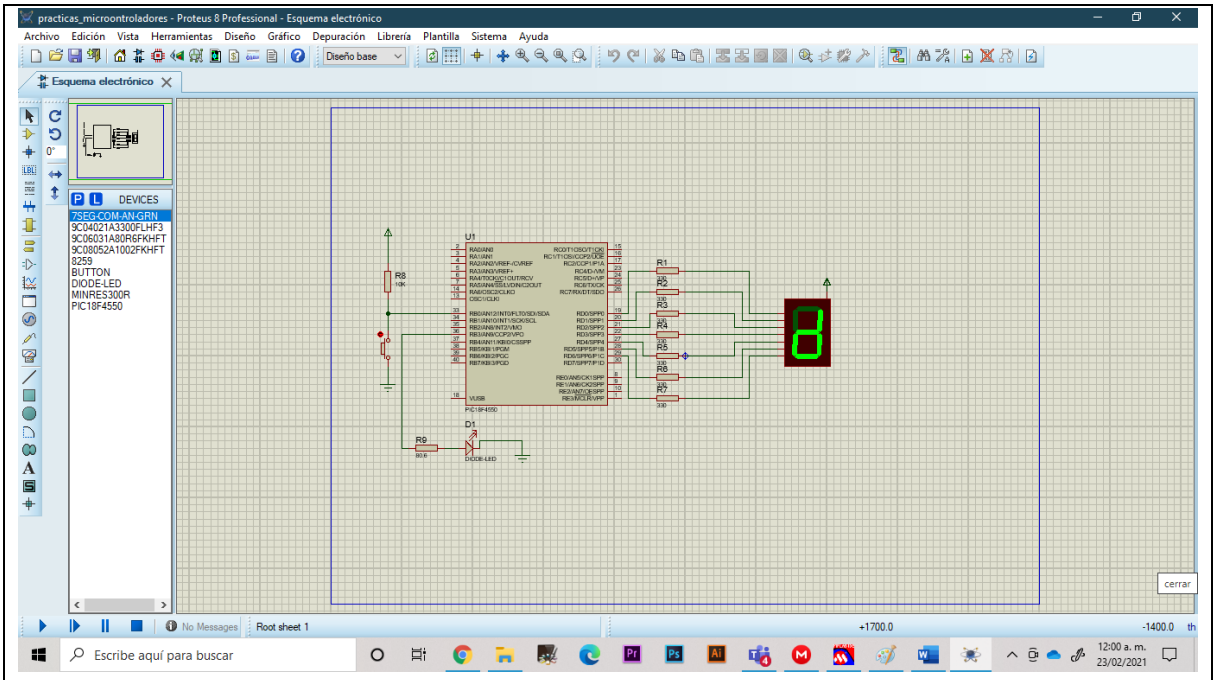

Imagen 13. Simulación en el software *Proteus*

Conclusiones

Con el desarrollo de la práctica 2 hemos podido aprender a utilizar nuevas instrucciones, además de que hemos podido comprender mejor el uso de las interrupciones y de nuevas directivas.

Link del vídeo: https://youtu.be/beQ5hOq6B-4

Bibliografía

Microchip PIC18F Instruction Set. (2021). Retrieved 9 February 2021, from http://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/

(2021). Retrieved 23 February 2021, from http://ww1.microchip.com/downloads/en/devicedoc/33014j.pdf