Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Electrónica

Microcontroladores

Materia

Práctica No. 4

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

Integrantes del equipo

Ingeniería en mecatrónica

Carrera

Ricardo Álvarez González

Docente

06 de abril de 2021, Primavera 2021

Fecha de entrega

Contenido

Objetivo

El objetivo de la práctica a realizar es poder entender el uso de los apuntadores y como podemos realizar una aplicación de estos utilizando un ejemplo real como lo es un contador básico en cuatro displays.

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
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).
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.
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).

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
Goto	[label] GOTO k	K -> PC <20:1>	2 palabras 2 ciclos	GOTO permite una ramificación incondicional donde sea dentro del rango de memoria de 2 MBytes. El valor de 20 bits de 'k' se carga dentro de PC<20:1>. GOTO es siempre una instrucción de dos ciclos.
Incf	[label] INCF f[,d[,a]	(f) + 1 -> dest	1 palabra 1 ciclo	El contenido del registro 'f' se incrementa
Movf	[label] MOVF f[,d[,a]	F -> dest	1 palabra 1 ciclo	El contenido del registro 'f' se mueve a un destino dependiendo del estatus de 'd'. Si 'd' es 0, el resultado se almacena en W. Si 'd' es 1, el resultado se almacena de regreso en 'f'.
Movff	[label] MOVFF f_s, f_d	(f_s) -> f_d	2 palabras 2 ciclos (3)	El contenido del registro fs se mueve al registro fd
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
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.
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
Xorlw	[label] XORLW k	(W).XOR.k -> w	1 palabra 1 ciclo	El contenido de W hace la operación XOR con la literal de 8 bits 'k'. El resultado se almacena en W.

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.

CBLOCK [expr]

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

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

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

Display de 7 segmentos

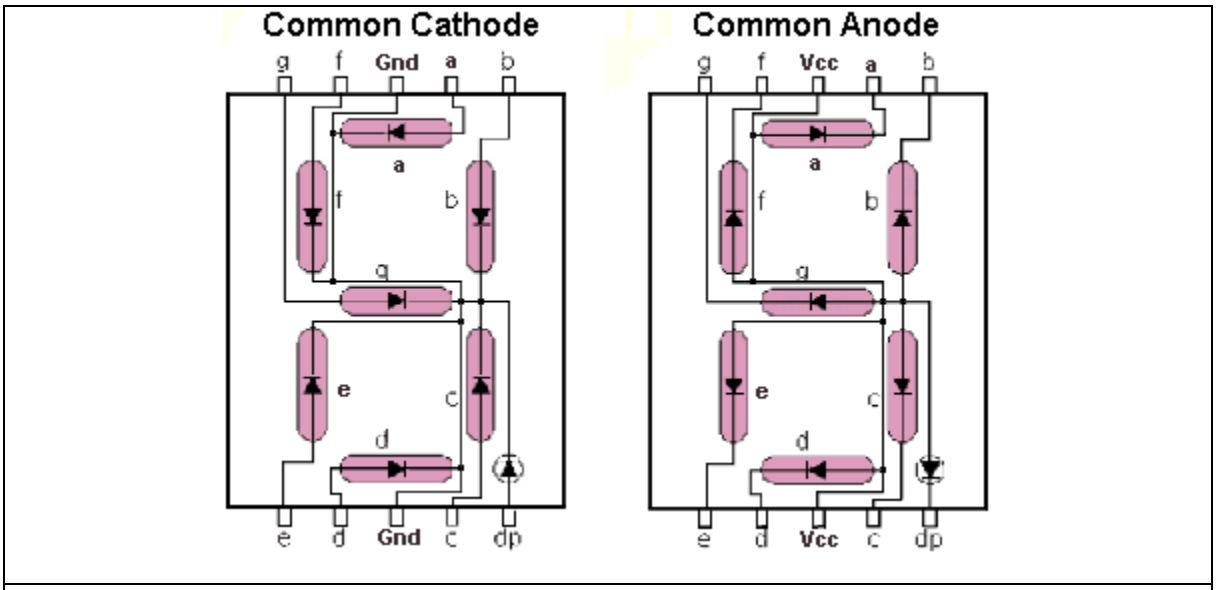


Imagen 1. Display de 7 segmentos de cátodo común (izquierda) y ánodo común (derecha).

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

Timer 0

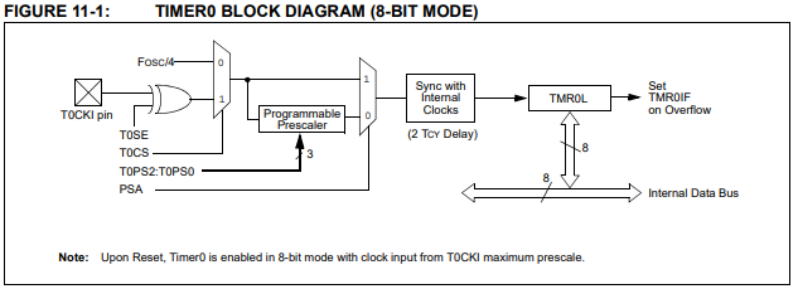
El módulo del Timer0 incorpora las siguientes características:

- Operación seleccionable por software como temporizador o contador, ambos en modo de 8 bits o 16 bits.
- Registros para lectura y escritura.
- Prescaler programable por software de 8 bits.
- Fuente seleccionable de reloj (interna o externa).
- Selección de borde para reloj externo.
- Interrupción en desbordamiento.

El registro T0CON controla todos los aspectos de las operaciones del módulo, incluyendo la selección de prescala. En ambos sirve de lectura y escritura.

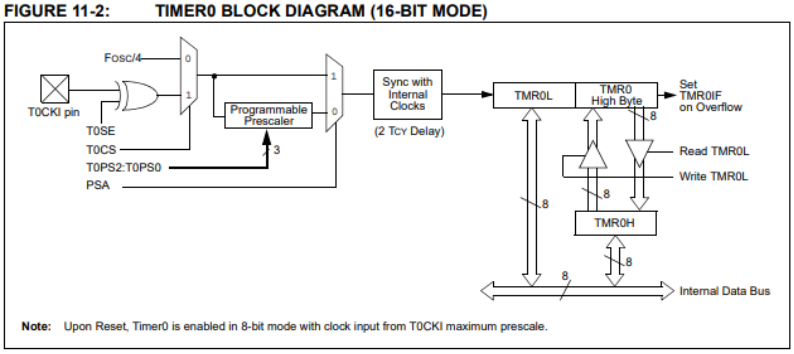
Un diagrama de bloques simplificado del módulo del timer0 en su modalidad de 8 bits y 16 bits se muestra en las imágenes siguientes.

FIGURE 11-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI maximum prescale.

FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI maximum prescale.

Imagen 3. Timer0 en su modalidad de 8 bits (Figure 11-1) y en su modalidad de 16 bits (Figure 11-2).

T0: Control de registro del Timer0

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Imagen 4. Funcionalidad de los bits del registro T0CON

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

Imagen 5. Función de acuerdo con los estados de los bits del registro T0CON

Operación del timer0

El Timer0 puede operar ya sea como temporizador o como contador, para seleccionar alguna de estas opciones se tiene que limpiar el bit T0CS (es decir, el bit 5 del registro T0CON). En el modo temporizador, el módulo incrementa en cada reloj por defecto a no ser que un valor de prescaler diferente sea seleccionado. Si el registro TMR0 está escrito, el incremento es inhibido para los dos siguientes ciclos de instrucción. El usuario puede solucionar esto escribiendo un valor ajustado para el registro del TMR0.

El modo contador es seleccionado al establecer el bit T0CS en 1. En modo contador, el timer0 incrementa ya sea en cada flanco de subida o bajada del pin RA4/T0CKI. El incremento en el flanco es determinado por el bit de selección de flanco de fuente del Timer0, T0SE (es decir, el bit 4 del registro T0CON); limpiando este bit se selecciona el flanco de subida.

Una señal externa de reloj puede ser usada para manejar el Timer0; como sea, se deben conocer ciertos requisitos para asegurarse que la señal externa de reloj esté sincronizada con la fase interna del reloj (Tosc). Existe un retraso entre la sincronización y el comienzo en el incremento el temporizador/contador.

Prescaler

Un contador de 8 bits está disponible como un prescaler para el modulo del Timer0. El prescaler no está directamente establecido para escribirse o leerse; su valor se establece por el PSA y T0PS2: los bits de T0PS0 (T0CON<3:0>) los cuales determinan la asignación del prescaler.

Interrupción del Timer0

La interrupción del TMR0 se genera cuando se desbordan los registros del TMR0 desde FFh hasta 00h en la modalidad de 8 bits, o desde FFFFh hasta 0000h en modalidad de 16 bits. Este desbordamiento establece el bit de bandera del TMR0IF. La interrupción puede ser enmascarada limpiando el bit TMR0IE (el bit 5 del registro INTCON). Antes de rehabilitar la interrupción, el bit TMR0IF debe ser limpiado en el software mediante una rutina de servicio de interrupción.

Desde que el Timer0 se apaga en el modo espera, la interrupción del TMR0 no puede despertar al procesador de ese modo.

Timer 2

El módulo del Timer2 incorpora las siguientes características:

- Temporizador de 8 bits y registros de periodo (TMR2 y PR2, respectivamente).
- Registros para lectura y escritura
- Prescaler programable por software (1:1, 1:4 y 1:16)
- Postscaler programable por software (1:1 hasta 1:16)
- Interrupción en TMR2 hasta PR2
- Uso opcional como reloj de cambio para el módulo MSSP

El módulo es controlado a través del registro T2CON el cuál habilita y deshabilita el temporizador y configura el prescaler y postscaler. El timer2 puede ser apagado limpiando el bit de control, TMR2ON (T2CON<2>), para minimizar el consumo de poder.

Un diagrama de bloques simplificado del módulo del timer2 se muestra en la figura 13-1.

FIGURE 13-1: TIMER2 BLOCK DIAGRAM

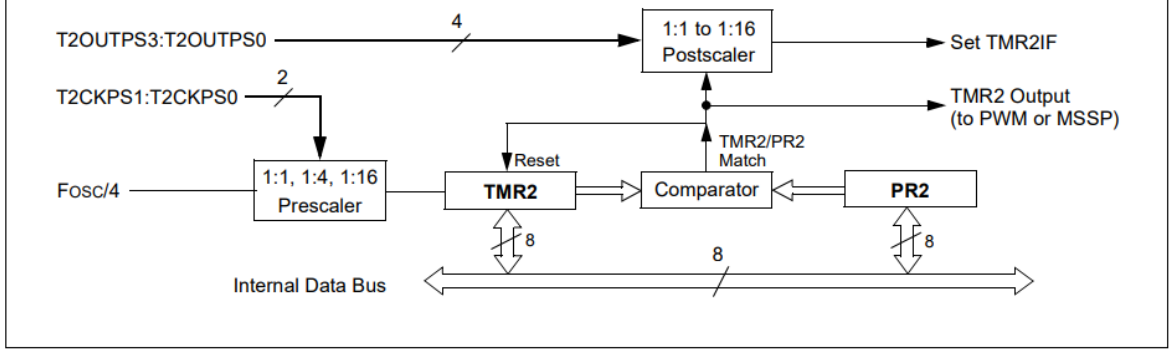


Imagen 6. Diagrama de bloques del Timer2

REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **T2OUTPS3:T2OUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Imagen 7. Registros de control del timer2

Operación del Timer2

En operación normal, TMR2 se incrementa de 00h en cada ciclo (Fosc/4). Un contador/prescaler de 2 bits en la entrada del reloj da una entrada directa, tiene la opción de dividir el prescaler entre 4 y 16. Estas formas son seleccionadas por el control de bits del prescaler T2CKPS1:T2CKPS0 (T2CON<1:0>). El valor del TMR2 se compara al del registro del periodo PR2 en cada ciclo de reloj. Cuando dos valores coinciden, el comparador genera una señal de comparación como salida del temporizador. Esta señal también reinicia el valor del TMR2 hasta 00h en el siguiente ciclo y conduce la salida del contador/postscaler.

Los registros TMR2 y PR2 permiten ambos ser escritos y leídos directamente. El registro TMR2 se limpia en un reinicio de cualquier dispositivo, mientras que el registro PR2 inicializa en FFh. Tanto el prescaler como el postscaler se limpian en las siguientes circunstancias:

- Una escritura en el registro TMR2
- Una escritura en el registro T2CON
- Reinicio de cualquier dispositivo (power on set, ¬mclr reset, watchdog timer reset o brow – out reset).

TMR2 no se limpia cuando el registro T2CON está escrito.

TABLE POINTER REGISTER (TBLPTR) y TABLE LATCH REGISTER (TABLAT)

El Table Latch (TABLAT) es un registro de 8 bits mapeado en el espacio de los SFR (Special Function Registers). El registro TABLAT es usado para guardar datos de 8 bits durante transferencia de datos entre la memoria de programa y los datos de la RAM.

El registro Table Pointer (TBLPTR) direcciona un byte dentro de la memoria de programa. El TBLPTR está compuesto de tres SFR: Table Pointer Upper Byte, Table Pointer High Byte y Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). Estos tres registros se unen para formar un apuntador de 22 bits. El orden bajo de 21 bits permite al dispositivo direccionar hasta 2 MB de espacio de memoria de programa. El bit 22 permite el acceso al ID del dispositivo, del usuario y la configuración de bits. El puntero de tabla, TBLPTR, es usado por las instrucciones TBLRD y TBLWT. Estas instrucciones pueden actualizar el TBLPTR en una de cuatro maneras de acuerdo con la tabla de operaciones. Estas operaciones se muestran en la tabla 6 – 1. Estas operaciones en el registro TBLPTR solamente afectan el orden bajo de 21 bits.

TABLE 6-1: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

Example	Operation on Table Pointer
TBLRD* TBLWT*	TBLPTR is not modified
TBLRD*+ TBLWT*+	TBLPTR is incremented after the read/write
TBLRD*- TBLWT*-	TBLPTR is decremented after the read/write
TBLRD+* TBLWT+*	TBLPTR is incremented before the read/write

FIGURE 6-3: TABLE POINTER BOUNDARIES BASED ON OPERATION

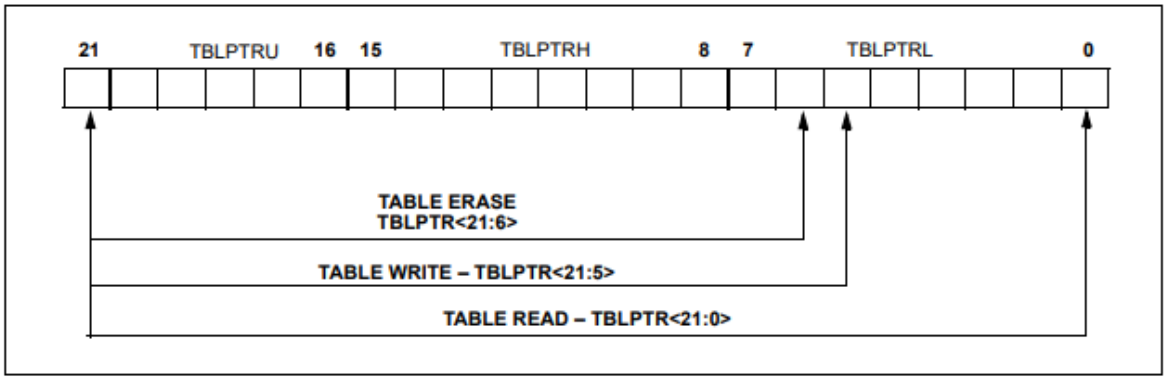


Imagen 8. Operaciones de apuntador de tabla (tabla 6 – 1) y límites de puntero de tabla basados en la operación (figura 6 – 3).

El registro TBLPTR es usado en lectura, escritura y borrado de la memoria flash del programa.

Cuando un TBLRD es ejecutado, todos los 22 bits del TBLPTR determinan cual byte es leído de la memoria del programa dentro de TABLAT.

Desarrollo práctico

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

Cantidad	Concepto
1	Tarjeta de desarrollo Miuva
1	Laptop
4	Display de 7 segmentos ánodo común
1	Software MPLAB v8.92
1	Datasheet del PIC 18F4550
7	Resistencia de 220
1	Resistencia de 330
1	Resistencia de 10k
4	Resistencia de 1k
4	Transistor BC547
2	Protoboard
1	Diodo LED
1	Pushbutton
20	Jumpers
1	Mt de alambre calibre 22

Lo primero que se realizó fue el montaje del circuito, donde se puede observar que se usa el puerto D para poder mostrar los números en cada display, mientras que el puerto A de la tarjeta de desarrollo miuva se utiliza para indicara que display cambiará el número que muestra.

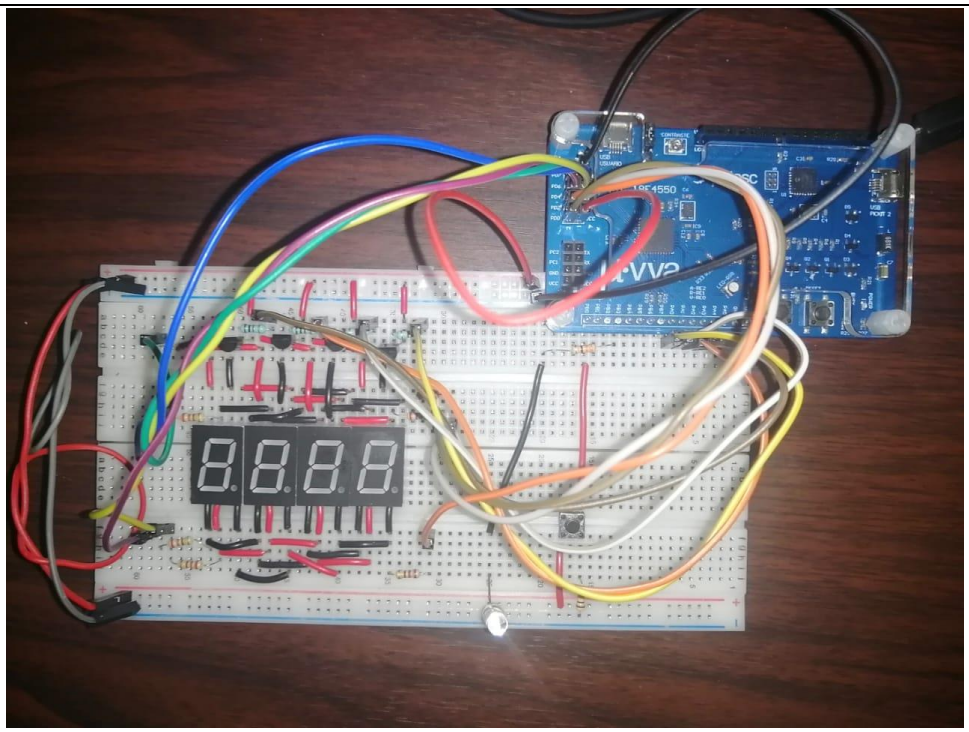


Imagen 9. Implementación del circuito para el desarrollo de la práctica

El código se explicará a continuación por pequeños segmentos para ahorrar el espacio utilizado, de todas formas, se mostrará más delante del documento para consultas.

En el fragmento de código que se muestra a continuación se declaran primero las directivas y se definen algunos bits de configuración, además de que en la parte inferior se definen las variables que se utilizarán en el código.

```
        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 ;ejemplo de definición de variables en RAM de acceso
        flags           ;banderas
        iun             ;índice de unidades
        cuni            ;código de 7 segmentos de unidades
        idec            ;índice de decenas
        cdec            ;código de 7 segmentos de decena
        icen            ;índice de centenas
        ccen            ;código de centenas
        iunimil         ;índice de unidades de millar
        cunimil         ;código de unidades de millar
        cont
        endc            ;fin de bloque de constantes
;Todo lo anterior son palabras de configuración
;*****************************************************************************asdasd
```

Imagen 10. Definición de directivas y de variables

En el siguiente fragmento del código, se muestra que se utiliza la directiva ***org*** para establecer el origen de las subrutinas subsecuentes.

En la parte de abajo se puede observar que se configura el oscilador interno a 4 MHz configurando el registro OSCCON (Oscilator Control Register), además de que se configuran los puertos como digitales y tanto el puerto D como el puerto A se configuran como salida limpiando los registros TRISD y TRISA, y por último se configura el bit 3 del puerto B como salida.

Mediante el registro INTCON configuramos algunas interrupciones externas (INT0, TMR0 y PEIE) asignando el valor hexadecimal 0xF0 a dicho registro. Por otra parte, se configura el TMR0 mediante los registros T0CON, TMR0H y TMR0L (se establece el prescaler y el valor de precarga).

```
;Reset vector
        ORG 0x0000
        bra     inicio
        org     0x08
        bra     RSI

;Inicio del programa principal
inicio  bsf     OSCCON, IRCF2, 0
        bsf     OSCCON, IRCF1, 0
        bcf     OSCCON, IRCF0, 0        ; 110, Oscilador interno a 4 MHz
        movlw   0x0F
        movwf   ADCON1, 0               ; Puertos digitales
        clrf    PORTD, 0
        clrf    TRISD, 0                ; Puerto D configurado como salida
        clrf    TRISA, 0                ; Puerto A configurado como salida
        bcf     TRISB, 3                ; RB3 salida
        movlw   0xF0                    ; 1111 0000
        movwf   INTCON, 0               ; Interrupciones externa INT0, TMR0 y PEIE

        movlw   0x95
        movwf   T0CON                   ; Timer 16 bits, preescaler *64
        movlw   0xE1
        movwf   TMR0H, 0
        movlw   0x7C
        movwf   TMR0L, 0                ; Valor de precarga para 1000ms a 4 MHz, preescaler 64 del timer0
```

Imagen 11. Configuración de registros

En este segmento del código, se configura el apuntador de la tabla (se limpia el registro TBLPTRL, se le asigna el valor 0x03 al registro TBLPTRH y por último se limpia el registro TBLPTRU, de manera que el valor de tblptr será 0x000300).

Por último, se configuran los registros del TMR2 (PR2 y T2CON), además de que se activa la interrupción del timer2 mediante el bit TMR2IE del registro PIE1.

Por último, en la parte inferior se puede observar que se limpian los registros *iun, idec, icen* y *iunimil* que serán los que se utilizarán para ser los índices de los valores.

```
clrf    TBLPTRL, 0
movlw   0x03
movwf   TBLPTRH, 0              ; El apuntador TBLPTRH se pone en dirección 0x03
clrf    TBLPTRU, 0              ; tblptr = 0x000300
movlw   0x05
movwf   PR2                     ; Timer2 period register
movlw   0x07
movwf   T2CON                   ;
bsf     PIE1, TMR2IE            ; Interrupción timer2 activa

clrf    iun, 0
clrf    idec, 0                 ; Iniciamos en cero
clrf    icen, 0
clrf    iunimil, 0
```

Imagen 12. Configuración del apuntador de tabla

En la subrutina *lee* se mueve el valor del registro *iun* al registro *TBLPTRL* (Table Pointer Low), se lee el valor del código del dígito y se mueve de *TABLAT* hacia *cuni,* y se repite el proceso para las decenas, centenas y unidades de millar.

```
lee     movff   iun, TBLPTRL            ; ajusta apuntador (unidades)
        tblrd   *                       ; lee la tabla sin modificar apuntador
        movff   TABLAT, cuni            ; Cuni tiene código 7 segmentos

        movff   idec, TBLPTRL           ; ajusta apuntador (decimales)
        tblrd   *                       ; Lee la tabla sin modificar apuntador
        movff   TABLAT, cdec            ; cdec tiene código 7 segmentos

        movff   icen, TBLPTRL           ; ajusta apuntador (centenas)
        tblrd   *                       ; lee la tabla sin modificar apuntador
        movff   TABLAT, ccen            ; Ccen tiene código 7 segmentos

        movff   iunimil, TBLPTRL
        tblrd   *
        movff   TABLAT, cunimil
```

Imagen 13. Subrutina *lee*

En la subrutina *loop* se mueve primero el valor 0x01 al puerto A, esto es para que se encienda el display de las unidades, por lo que también se observa que se mueve el valor del registro *cuni* hacia el puerto D. El puerto A en este caso nos sirve para indicar que display se va a encender, así que utilizamos ese mismo recurso para los otros 3 displays restantes. Los valores que se asignan a dichos puertos coinciden en sistema binario con los displays que se encenderán (0x01 = 0001, 0x02=0010, 0x04=0100, 0x08=1000). Al final de esta secuencia, se verificará si el bit 0 de bandera es 1 (esto para saber si ya han pasado 1000 ms), si es así, entonces se apagará el bit de bandera (imagen 15) y se incrementará una unidad, de lo contrario, se regresará al inicio de la subrutina *loop*.

```
loop    movlw   0x01
        movwf   PORTA,0                 ; encendemos display unidades
        movff   cuni, PORTD             ; Mueve el valor del código de las unidades al puerto D (Codigo UNIdades)
        call    delay                   ; Timer2 establecerá el tiempo que esté encendido el display

        movlw   0x02
        movwf   PORTA, 0                ; encendemos display decenas
        movff   cdec, PORTD             ; Mueve el valor del código de las decenas (Codigo DECenas)
        call    delay                   ; Timer2 establecerá el tiempo que esté encendido el display

        movlw   0x04
        movwf   PORTA, 0                ; encendemos display centenas
        movff   ccen, PORTD             ;Mueve el valor del código de las centenas al puerto D (Código CENtenas)
        call    delay

        movlw   0X08                    ; encendemos display unidades de milar
        movwf   PORTA, 0
        movff   cunimil, PORTD
        call    delay
        btfss   flags, 0, 0
        bra     loop
```

Imagen 14. Subrutina *loop*

Una vez que termina la subrutina *loop*, si ha pasado el tiempo de 1000 ms se procede a incrementar el registro unidades para después mover su valor al registro *Working register* y hacer la operación XOR con el valor 0x0a (9 decimal). Si ambos valores son iguales (lo que indicaría que las unidades han llegado a 9 por lo que se tendría que avanzar una decena) entonces se verifica el bit Z del registro *STATUS*, el cuál será 1 si la operación XOR es 1 (será 1 cuando ambos valores de los registros son iguales). Si aún no llegan a 9 las unidades, ese bit permanecerá en 0 por lo que se regresará a la subrutina *lee*, de lo contrario, se limpiará el registro *iun* y se procederá a incrementar el valor de las

decenas y se hará la misma evaluación (verificar si está en 9). Ese proceso se repite hasta llegar a las unidades de millar.

```
bcf     flags, 0, 0             ; ¿Ya transcurrieron 500 ms?
incf    iun, F, 0               ; Se incrementan los índices de unidades
movf    iun, W, 0               ; Se mueve el valor de iun a wreg
xorlw   0x0a
btfss   STATUS, Z, 0            ; verifica límite de tabla (si se hace 0 con la operación quiere decir que llegó al l
bra     lee
clrf    iun, 0

incf    idec, F, 0
movf    idec, W, 0
xorlw   0x0a
btfss   STATUS, Z, 0
bra     lee
clrf    idec, 0

incf    icen, F, 0
movf    icen, W, 0
xorlw   0x0a
btfss   STATUS, Z, 0
bra     lee
clrf    icen, 0

incf    iunimil, F, 0
movf    icen, W, 0
xorlw   0x0a
btfss   STATUS, Z, 0
bra     lee
clrf    iunimil, 0
goto    lee
```

Imagen 15. Tiempo de 500 ms e incremento de valores

En la última parte se tiene la rutina de servicio de interrupción. En la subrutina *RSI* se verificará si el bit de interrupción está encendido, si lo está se limpiará, de lo contrario irá a la subrutina *SINT0* y verificará si la interrupción del bit INT0IF está encendida, de lo contrario verificará la última interrupción que se encuentra en la subrutina *ST2*, la cuál es la del timer2, del bit *TMR2IF* del registro *PIR1*.

```
;*****RUTINA DE SERVICIO DE INTERRUPCIÓN************************************************
RSI     btfss   INTCON, TMR0IF
        bra     SINT0
        bcf     INTCON, TMR0IF
        movlw   0xC2
        movwf   TMR0H, 0
        movlw   0xF7
        movwf   TMR0L, 0                ; valor de precarga para 1000 ms a 4 MHz preescaler 64
        bsf     flags, 0                ; monitor interrupción del timer 0
        retfie
SINT0   btfss   INTCON, INT0IF
        bra     ST2
        bcf     INTCON, INT0IF          ; apaga bit de bandera
        btg     PORTB, 3                ; puerto monitor de interrupción
        retfie
ST2     bcf     PIR1, TMR2IF            ;Peripheral Interrupts Request 1 & Timer2 Interrupt flag bit
        bsf     flags, 2                ; Bandera monitor del timer2
        retfie
;*******************************************************************************************
delay   btfss   flags, 2
        bra     delay
        bcf     flags, 2
        return

;*******************************************************************************************
        org     0x300                   ; DB Directiva que Define Byte
        DB      0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xB8, 0x80, 0x98, 0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e
        END
```

Imagen 16. Rutina de servicio de interrupción y subrutina *delay*

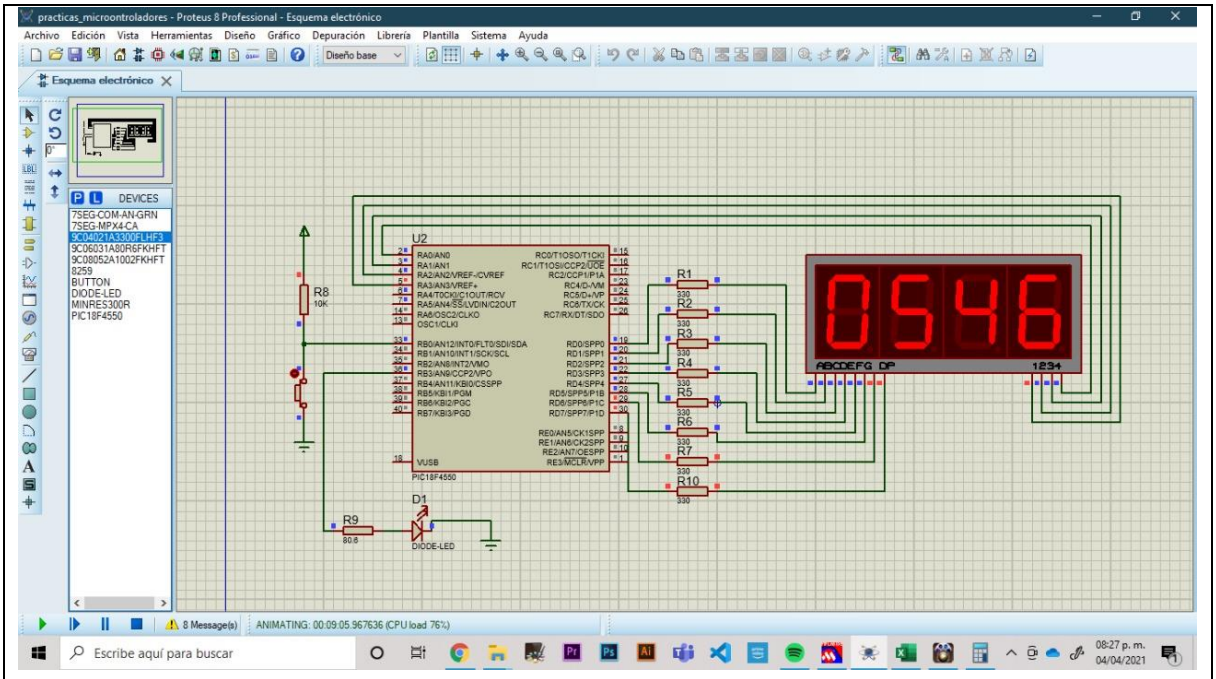


Imagen 17. Simulación del circuito en el software *Proteus 8*

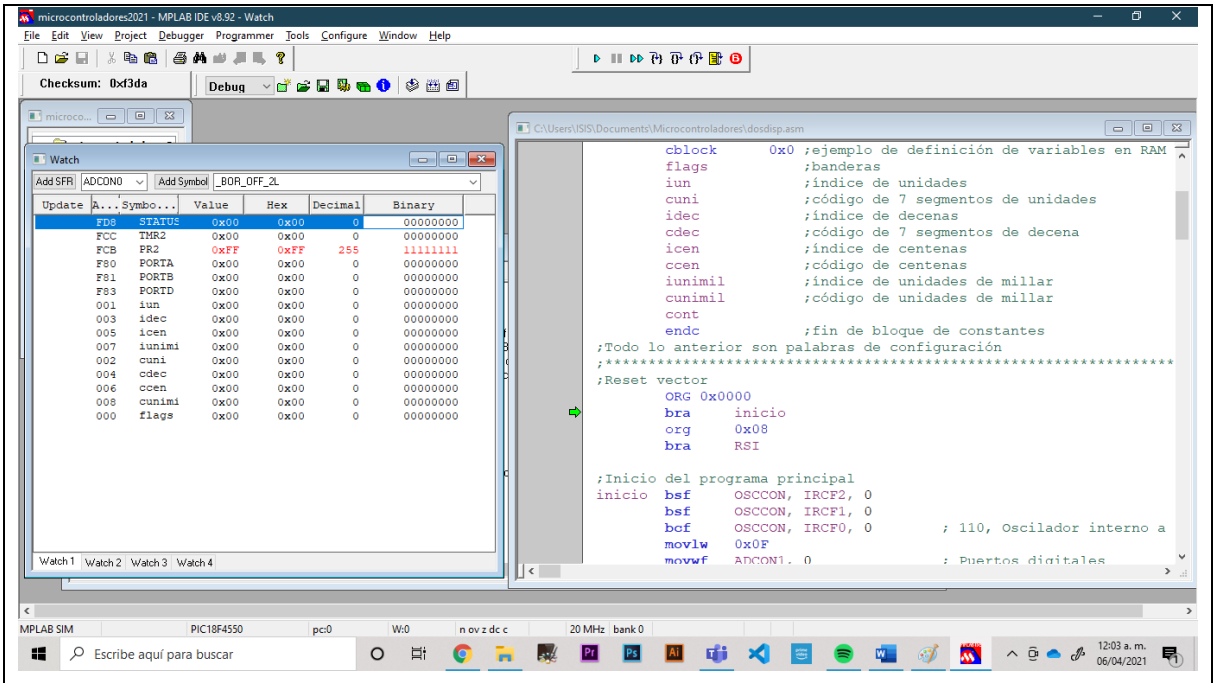


Imagen 18. Simulación en MPLAB IDE v8.92

Para la implementación del circuito en la Protoboard se tuvo que realizar el siguiente circuito por cada display conectado.

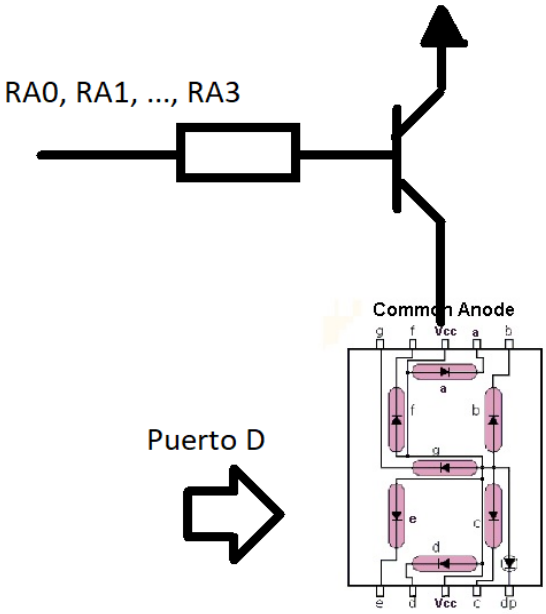


Imagen 19. Conexión del display de ánodo común con el puerto A y el puerto D

Código completo en ASM

```
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     ;ejemplo de definición de variables en RAM de acceso
                flags                   ;banderas
                iun                             ;índice de unidades
                cuni                    ;código de 7 segmentos de unidades
                idec                    ;índice de decenas
                cdec                    ;código de 7 segmentos de decena
                icen                    ;índice de centenas
                ccen                    ;código de centenas
                iunimil                 ;índice de unidades de millar
                cunimil                 ;código de unidades de millar
                cont
```

```
                endc                    ;fin de bloque de constantes
;Todo lo anterior son palabras de configuración
;*****************************************************************************asda
sd
;Reset vector
                ORG 0x0000
                bra     inicio
                org             0x08
                bra             RSI

;Inicio del programa principal
inicio  bsf             OSCCON, IRCF2, 0
                bsf     OSCCON, IRCF1, 0
                bcf             OSCCON, IRCF0, 0        ; 110, Oscilador interno a 4 MHz
                movlw   0x0F
                movwf   ADCON1, 0                       ; Puertos digitales
                clrf    PORTD, 0
                clrf    TRISD, 0                        ; Puerto D configurado como salida
                clrf    TRISA, 0                        ; Puerto A configurado como salida
                bcf     TRISB, 3                        ; RB3 salida
                movlw   0xF0                            ; 1111 0000
                movwf   INTCON, 0                       ; Interrupciones externa INT0,
TMR0 y PEIE

                movlw   0x95
                movwf   T0CON                           ; Timer 16 bits, preescaler *64
                movlw   0xE1
                movwf   TMR0H, 0
                movlw   0x7C
                movwf   TMR0L, 0                        ; Valor de precarga para 1000ms a
4 MHz, preescaler 64 del timer0

                clrf    TBLPTRL, 0
                movlw   0x03
                movwf   TBLPTRH, 0                      ; El apuntador TBLPTRH se pone en
dirección 0x03
                clrf    TBLPTRU, 0                      ; tblptr = 0x000300
                movlw   0x05
                movwf   PR2                                     ; Timer2 period register
                movlw   0x07
                movwf   T2CON                           ;
                bsf             PIE1, TMR2IE            ; Interrupción timer2 activa

                clrf    iun, 0
                clrf    idec, 0                         ; Iniciamos en cero
                clrf    icen, 0
                clrf    iunimil, 0
; Subrutina Lee donde se pasa el valor de "iun" al apuntador "TBLPTRL" y se lee su valor
```

```
; Se mueve el valor de donde el apuntador apuntó (donde ahora está TABLAT) al código de las
unidades y se repite el proceso
; con los decimales (Lo que hace la sr lee es que va a conseguir los códigos para mostrar los
números en los displays)
lee             movff   iun, TBLPTRL            ; ajusta apuntador (unidades)
                tblrd   *                                       ; lee la tabla sin modificar
apuntador
                movff   TABLAT, cuni            ; Cuni tiene código 7 segmentos

                movff   idec, TBLPTRL           ; ajusta apuntador (decimales)
                tblrd   *                                       ; Lee la tabla sin modificar
apuntador
                movff   TABLAT, cdec            ; cdec tiene código 7 segmentos

                movff   icen, TBLPTRL           ; ajusta apuntador (centenas)
                tblrd   *                                       ; lee la tabla sin modificar
apuntador
                movff   TABLAT, ccen            ; Ccen tiene código 7 segmentos

                movff   iunimil, TBLPTRL
                tblrd   *
                movff   TABLAT, cunimil
; Inicio del programa principal
loop    movlw   0x01
                movwf   PORTA,0                                 ; encendemos display
unidades
                movff   cuni, PORTD                     ; Mueve el valor del código de las
unidades al puerto D (Codigo UNIdades)
                call    delay                           ; Timer2 establecerá el tiempo que
esté encendido el display

                movlw   0x02
                movwf   PORTA, 0                        ; encendemos display decenas
                movff   cdec, PORTD                     ; Mueve el valor del código de las
decenas (Codigo DECenas)
                call    delay                           ; Timer2 establecerá el tiempo que
esté encendido el display

                movlw   0x04
                movwf   PORTA, 0                        ; encendemos display centenas
                movff   ccen, PORTD                     ;Mueve el valor del código de las
centenas al puerto D (Código CENtenas)
                call    delay

                movlw   0X08                            ; encendemos display unidades de
milar
                movwf   PORTA, 0
                movff   cunimil, PORTD
```

```
                call    delay
                btfss   flags, 0, 0
                bra             loop
; Programa principal de loop hasta acá jejeps
                bcf             flags, 0, 0                             ; ¿Ya transcurrieron 500
ms?
                incf    iun, F, 0                       ; Se incrementan los índices de unidades
                movf    iun, W, 0                               ; Se mueve el valor de iun a wreg
                xorlw   0x0a
                btfss   STATUS, Z, 0                    ; verifica límite de tabla (si se hace 0 con la
operación quiere decir que llegó al límite que pusimos)
                bra             lee
                clrf    iun, 0

                incf    idec, F, 0
                movf    idec, W, 0
                xorlw   0x0a
                btfss   STATUS, Z, 0
                bra             lee
                clrf    idec, 0

                incf    icen, F, 0
                movf    icen, W, 0
                xorlw   0x0a
                btfss   STATUS, Z, 0
                bra             lee
                clrf    icen, 0

                incf    iunimil, F, 0
                movf    icen, W, 0
                xorlw   0x0a
                btfss   STATUS, Z, 0
                bra             lee
                clrf    iunimil, 0
                goto    lee
;*******************************************************************************
*****


;*****RUTINA DE SERVICIO DE
INTERRUPCIÓN*********************************************
RSI             btfss   INTCON, TMR0IF
                bra             SINT0
                bcf             INTCON, TMR0IF
                movlw   0xC2
                movwf   TMR0H, 0
                movlw   0xF7
```

```
                movwf  TMR0L, 0                         ; valor de precarga para 1000 ms a
4 MHz preescaler 64
                bsf             flags, 0                ; monitor interrupción del timer 0
                retfie
SINT0   btfss   INTCON, INT0IF
                bra             ST2
                bcf             INTCON, INT0IF          ; apaga bit de bandera
                btg             PORTB, 3                        ; puerto monitor de
interrupción
                retfie
ST2             bcf             PIR1, TMR2IF            ;Peripheral Interrupts Request 1 &
Timer2 Interrupt flag bit
                bsf             flags, 2                ; Bandera monitor del timer2
                retfie
;*************************************************************************
*****
;espera btfss    flags, 0                        ; ¿pasaron 500 ms?
;               bra             espera
;               bcf     flags, 0
;               return
;*************************************************************************
*****
delay   btfss   flags, 2
                bra             delay
                bcf             flags, 2
                return

;*************************************************************************
*****
                org             0x300                   ; DB Directiva que Define
Byte
                DB              0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xB8, 0x80, 0x98, 0x88,
0x83, 0xc6, 0xa1, 0x86, 0x8e
                END
```

Conclusiones

La realización de esta práctica nos ha permitido poder utilizar otra herramienta más que nos ofrece el lenguaje ASM, la cuál es poder almacenar variables en cierta dirección para poder acceder a ellos de manera más ordenada mediante apuntadores, lo cuál es útil cuando requerimos almacenar muchos datos que son para un mismo objetivo.

Link del vídeo: https://youtu.be/s7O55wPHHNQ

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