	Laboratorio de Microcomputadoras

Profesor(a): Dra. Lourdes Angelica Quiñonez Juárez

Asignatura: Laboratorio de Microcomputadoras

Grupo Laboratorio: 5

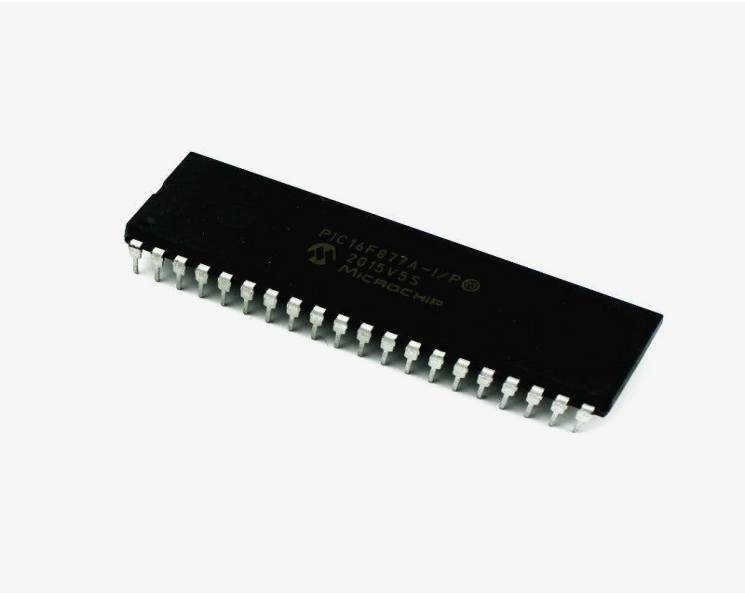
Grupo Teoría: 3

No. de Práctica(s): 2

Integrante(s): Hernández Diaz Sebastián

Semestre: 2025-2

Fecha de entrega: 3 de marzo del 2025



Hernández Diaz Sebastian

Laboratorio de Microcomputadoras

Práctica No. 2

Sistema mínimo microcontrolador PIC16F877

Objetivo. Conocer la estructura y características de la tarjeta que se dispone en el laboratorio, el software de comunicación, aplicaciones con puertos paralelos trabajando como salida y la ejecución de un programa en tiempo real.

Desarrollo. Para cada uno de los siguientes ejercicios, realizar los programas solicitados y comprobar el funcionamiento de ellos.

1. **Escribir, comentar e indicar que hace el siguiente programa.**

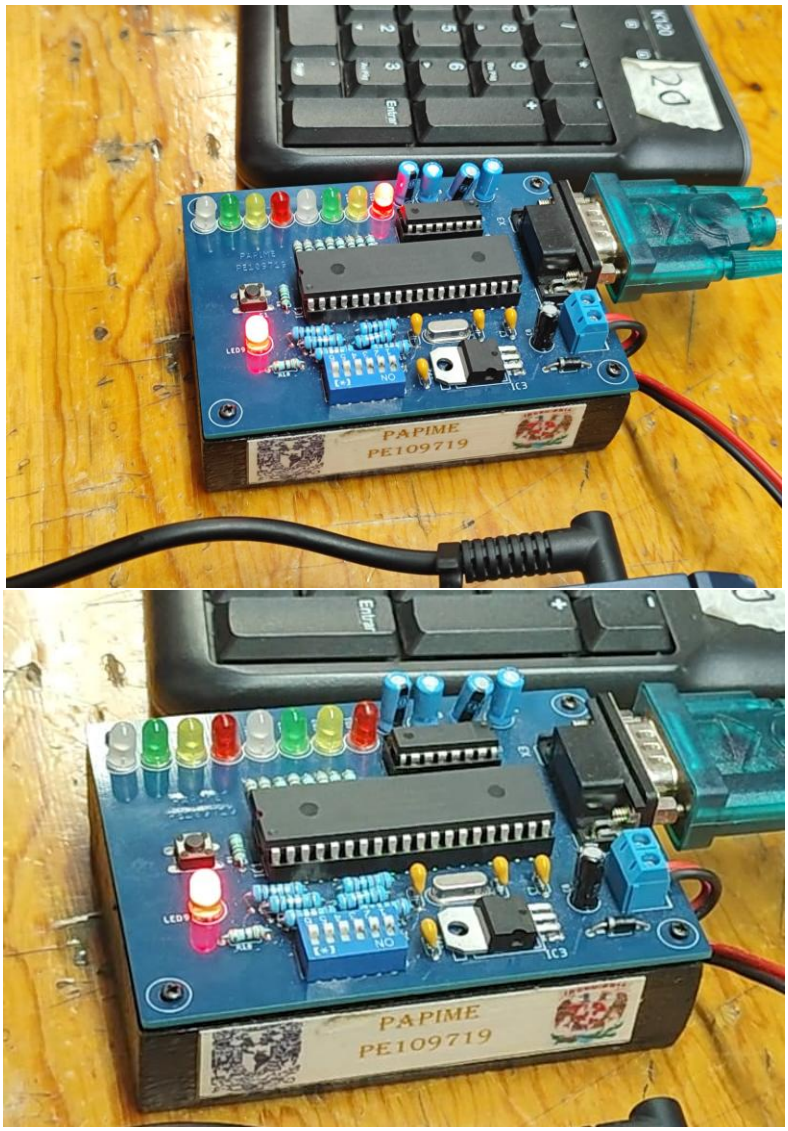
```
    processor 16f877       ; Define el procesador
    include <p16f877.inc>

    contador equ h'20'  ; Define la dirección de memoria 0x20 para la variable "contador"
    valor1   equ h'21'  ; Define la dirección de memoria 0x21 para la variable "valor1"
    valor2   equ h'22'  ; Define la dirección de memoria 0x22 para la variable "valor2"
    valor3   equ h'23'  ; Define la dirección de memoria 0x23 para la variable "valor3"
    cte1     equ 20h    ; Define una constante con valor 0x20
    cte2     equ 50h    ; Define una constante con valor 0x50
    cte3     equ 60h    ; Define una constante con valor 0x60
                        ; Estos valores son para un retardo de medio segundo

    org 0             ; Establece el origen en la dirección 0x0000
    goto inicio       ; Salta a la etiqueta "inicio"

    org 5             ; Establece la dirección de inicio del programa en 0x0005

inicio:
    bsf STATUS,5      ; Cambia al banco de registros 1
    BCF STATUS,6      ; Asegura que estamos en el banco correcto (banco 1)
    MOVLW H'0'        ; Carga el valor 0 en el registro W
    MOVWF TRISB       ; Configura todos los pines del puerto B como salida
    BCF STATUS,5      ; Regresa al banco 0
    clrf PORTB        ; Limpia el puerto B (pone todos los pines en 0)
```

```
loop2:
    bsf PORTB,0       ; Activa el bit 0 del puerto B
    call retardo      ; Llama a la funcion de retardo
    bcf PORTB,0       ; Apaga el bit 0 del puerto B
    call retardo      ; Llama a la funcion de retardo
    goto loop2        ; Repite el ciclo de encender y apagar el bit 0 del puerto B

retardo:
    movlw ctel        ; Carga el valor de la constante ctel en W
    movwf valor1      ; Guarda el valor en la variable valor1
tres:
    movlw cte2        ; Carga el valor de la constante cte2 en W
    movwf valor2      ; Guarda el valor en la variable valor2
dos:
    movlw cte3        ; Carga el valor de la constante cte3 en W
    movwf valor3      ; Guarda el valor en la variable valor3
uno:
    decfsz valor3     ; Decrementa valor3 y si llega a 0, salta la siguiente instrucción
    goto uno          ; Si valor3 no es 0, repite el ciclo

    decfsz valor2     ; Decrementa valor2 y si llega a 0, salta la siguiente instrucción
    goto dos          ; Si valor2 no es 0, repite el ciclo

    decfsz valor1     ; Decrementa valor1 y si llega a 0, salta la siguiente instrucción
    goto tres         ; Si valor1 no es 0, repite el ciclo

    return

    END
```

Este programa configura el puerto B del PIC16F877 como salida y hace que el bit 0 de dicho puerto se encienda y se apague repetidamente con un retardo entre cada cambio de estado. El retardo se logra con una función basada en decrementos de valores almacenados en registros de memoria, lo que genera un ciclo de espera antes de regresar a la ejecución principal que en este caso dados los valores el retardo es de medio segundo.

2. **Ensamblar y cargar el programa anterior en memoria del microcontrolador.**

   El programa anterior cargado y ejecutado en el microcontrolador se ve de la siguiente manera:



En este caso se puede ver que el led se apaga y se prende como debería de ser, también se tomaron los tiempos y fueron los correctos, el led se encendía y se apagaba cada medio segundo como fue solicitado.

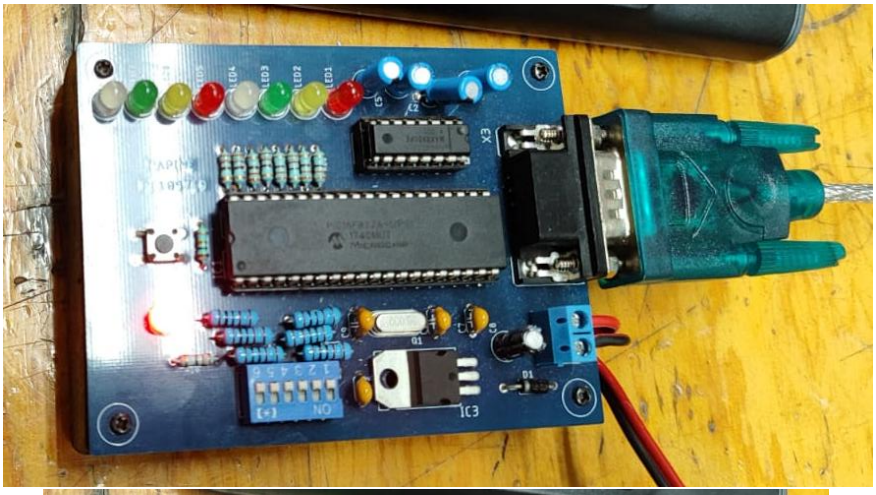
3. **Modificar el programa anterior, para que ahora se actualice el contenido de todos los bits del puerto B y se genere una rutina de retardo de un segundo.**

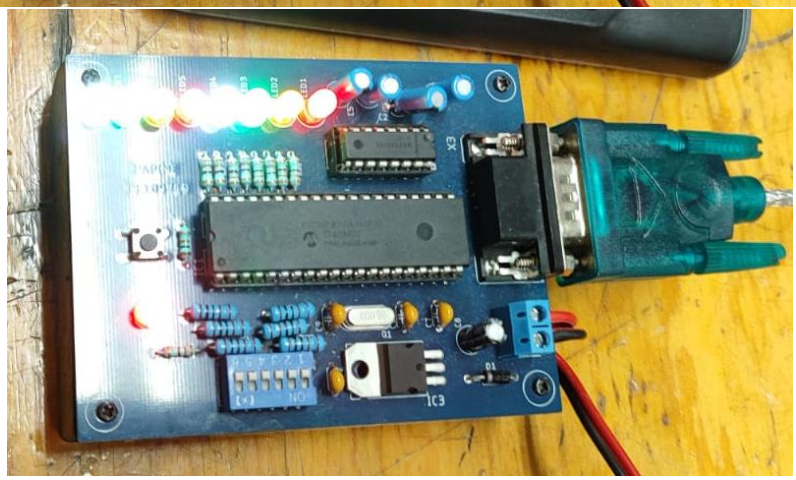
```
    processor 16f877       ; Define el procesador PIC16F877
    include <p16f877.inc>

    contador equ h'20'  ; Define la dirección de memoria 0x20 para la variable "contador"
    valor1   equ h'21'  ; Define la dirección de memoria 0x21 para la variable "valor1"
    valor2   equ h'22'  ; Define la dirección de memoria 0x22 para la variable "valor2"
    valor3   equ h'23'  ; Define la dirección de memoria 0x23 para la variable "valor3"
    cte1     equ 40h    ; Define una constante con valor 0x40
    cte2     equ 100h    ; Define una constante con valor 0x100
    cte3     equ 120h    ; Define una constante con valor 0x120
                        ; Estos valores dan un retardo de 1 segundo

    org 0             ; Establece el origen en la dirección 0x0000
    goto inicio       ; Salta a la etiqueta "inicio"

    org 5             ; Establece la dirección de inicio del programa en 0x0005

inicio:
    bsf STATUS,5      ; Cambia al banco de registros 1
    BCF STATUS,6      ; Asegura que estamos en el banco correcto (banco 1)
    MOVLW H'0'        ; Carga el valor 0 en el registro W
    MOVWF TRISB       ; Configura todos los pines del puerto B como salida
    BCF STATUS,5      ; Regresa al banco 0
    clrf PORTB        ; Limpia el puerto B (pone todos los pines en 0)

loop2:
    movlw h'ff'       ; Carga el valor 0xFF en el registro W
    movwf PORTB       ; Asigna el valor al puerto B (enciende todos los LEDs)
    call retardo      ; Llama a la subrutina de retardo

    clrf PORTB        ; Limpia el puerto B (apaga todos los LEDs)
    call retardo      ; Llama a la subrutina de retardo

    goto loop2        ; Repite el ciclo de encendido y apagado del puerto B

retardo:
    movlw cte1        ; Carga el valor de la constante cte1 en W
    movwf valor1      ; Guarda el valor en la variable valor1
tres:
    movlw cte2        ; Carga el valor de la constante cte2 en W
    movwf valor2      ; Guarda el valor en la variable valor2
dos:
    movlw cte3        ; Carga el valor de la constante cte3 en W
    movwf valor3      ; Guarda el valor en la variable valor3
uno:
    decfsz valor3     ; Decrementa valor3 y si llega a 0, salta la siguiente instrucción
    goto uno          ; Si valor3 no es 0, repite el ciclo

    decfsz valor2     ; Decrementa valor2 y si llega a 0, salta la siguiente instrucción
    goto dos          ; Si valor2 no es 0, repite el ciclo

    decfsz valor1     ; Decrementa valor1 y si llega a 0, salta la siguiente instrucción
    goto tres         ; Si valor1 no es 0, repite el ciclo

    return

    END
```

Este programa configura el puerto B del PIC16F877 como salida y hace que todos los bits del puerto B se enciendan (0xFF) y luego se apaguen (0x00)

de manera repetitiva, creando un efecto de parpadeo en los LEDs conectados al puerto B. El retardo entre cada cambio de estado se logra con una subrutina que utiliza decrementos de registros para generar un tiempo de espera antes de continuar con la ejecución.

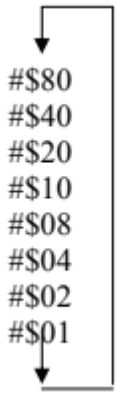
El código en ejecución es el siguiente:





Como se puede observar todos los leds prendieron de la forma solicitado y también se apagaban y prendían en el tiempo solicitado de 1 segundo.

4. **Realizar un programa que muestre la siguiente secuencia en el puerto B con retardos de 1⁄2 segundo.**



Secuencia:

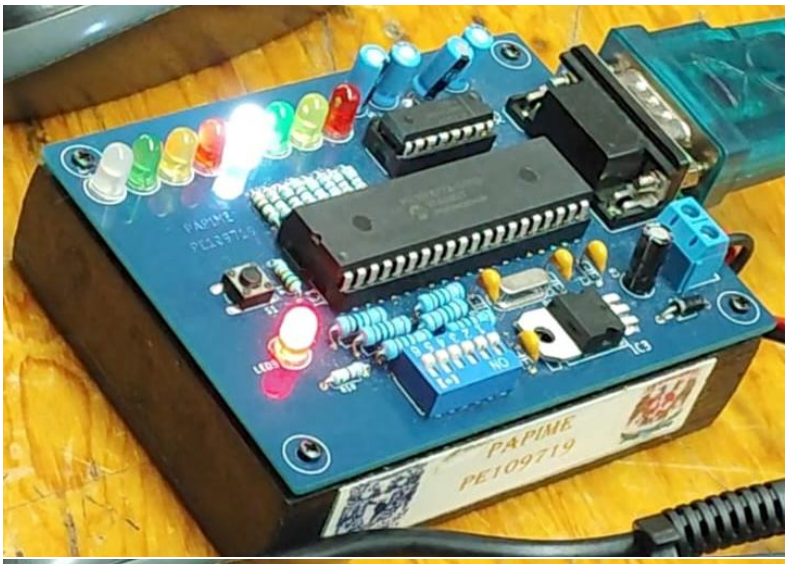
```
    processor 16f877       ; Define el procesador PIC16F877
    include <p16f877.inc>

    contador equ h'20'  ; Define la dirección de memoria 0x20 para la variable "contador"
    valor1   equ h'21'  ; Define la dirección de memoria 0x21 para la variable "valor1"
    valor2   equ h'22'  ; Define la dirección de memoria 0x22 para la variable "valor2"
    valor3   equ h'23'  ; Define la dirección de memoria 0x23 para la variable "valor3"
    cte1     equ 20h    ; Define una constante con valor 0x20
    cte2     equ 50h    ; Define una constante con valor 0x50
    cte3     equ 60h    ; Define una constante con valor 0x60
                        ; Estos valores dan un retardo de medio segundo

    org 0             ; Establece el origen en la dirección 0x0000
    goto inicio       ; Salta a la etiqueta "inicio"

    org 5             ; Establece la dirección de inicio del programa en 0x0005

inicio:
    bsf STATUS,5      ; Cambia al banco de registros 1
    BCF STATUS,6      ; Asegura que estamos en el banco correcto (banco 1)
    MOVLW H'0'        ; Carga el valor 0 en el registro W
    MOVWF TRISB       ; Configura todos los pines del puerto B como salida
    BCF STATUS,5      ; Regresa al banco 0
    clrf PORTB        ; Limpia el puerto B (pone todos los pines en 0)

loop2:
    movlw h'80'       ; Carga en W el valor 0x80 (10000000 en binario)
    movwf PORTB       ; Asigna el valor al puerto B (enciende solo el bit más significativo)
    call retardo      ; Llama a la funcion de retardo
    movwf contador    ; Guarda en "contador" el valor de W
```

```
corrimiento:
    rrf PORTB, 1      ; Desplaza los bits del puerto B una posición a la derecha
    call retardo      ; Llama a la funcion de retardo
    decf contador, 1  ; Decrementa el valor de "contador"
    btfss STATUS, 2   ; Si la bandera Z está en 1, significa que "contador" llegó a 0, entonces salta la siguiente línea
    goto corrimiento  ; Si "contador" no es cero, sigue desplazando el bit
    goto loop2        ; Reinicia la secuencia

retardo:
    movlw ctel        ; Carga el valor de la constante ctel en W
    movwf valor1      ; Guarda el valor en la variable valor1
tres:
    movlw cte2        ; Carga el valor de la constante cte2 en W
    movwf valor2      ; Guarda el valor en la variable valor2
dos:
    movlw cte3        ; Carga el valor de la constante cte3 en W
    movwf valor3      ; Guarda el valor en la variable valor3
uno:
    decfsz valor3     ; Decrementa valor3 y si llega a 0, salta la siguiente instrucción
    goto uno          ; Si valor3 no es 0, repite el ciclo

    decfsz valor2     ; Decrementa valor2 y si llega a 0, salta la siguiente instrucción
    goto dos          ; Si valor2 no es 0, repite el ciclo

    decfsz valor1     ; Decrementa valor1 y si llega a 0, salta la siguiente instrucción
    goto tres         ; Si valor1 no es 0, repite el ciclo

    return

    END
```

Este programa configura el puerto B del PIC16F877 como salida y genera un efecto de desplazamiento de un bit hacia la derecha en el puerto B, comenzando con el bit más significativo encendido (10000000 en binario). Luego, el programa va desplazando este bit hacia la derecha con la instrucción rrf PORTB,1 (Rotate Right File Register), hasta que el bit desaparece. Después de completar un ciclo, el proceso se repite. Se usa una subrutina de retardo para hacer visible el desplazamiento del bit en hardware, probablemente con LEDs conectados al puerto B.

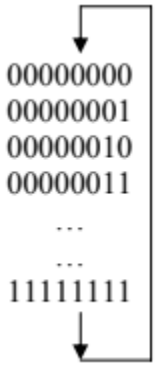
El codigo en ejecucion es el siguiente:

Hernández Diaz Sebastian



Como se puede ver en las imágenes la secuencia solicitada se muestra en el encendido de los leds y se ve como estos se van recorriendo al apagar y encenderse de manera correcta.

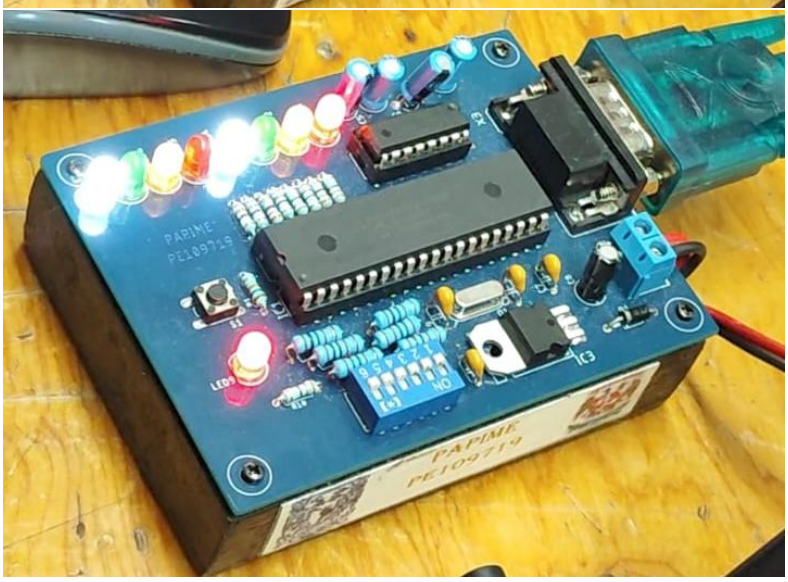
5. **Realizar un programa que muestre un contador binario por el puerto paralelo B, desde su valor mínimo B'00000000' hasta el máximo B'11111111' y se repita nuevamente el contador; usar retardos de 1⁄2 segundo.**

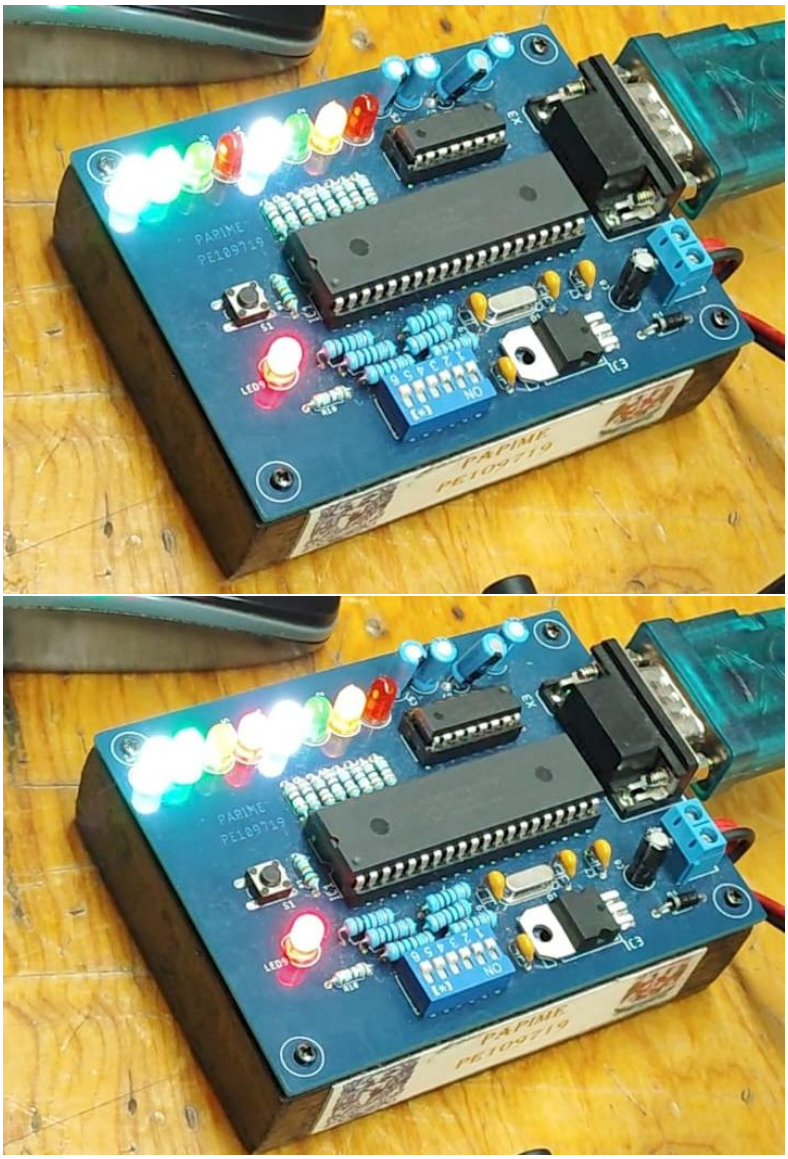


```
    processor 16f877       ; Define el procesador PIC16F877
    include <p16f877.inc>

    contador equ h'20'  ; Define la dirección de memoria 0x20 para la variable "contador"
    valor1   equ h'21'  ; Define la dirección de memoria 0x21 para la variable "valor1"
    valor2   equ h'22'  ; Define la dirección de memoria 0x22 para la variable "valor2"
    valor3   equ h'23'  ; Define la dirección de memoria 0x23 para la variable "valor3"
    cte1     equ 20h    ; Define una constante con valor 0x20
    cte2     equ 50h    ; Define una constante con valor 0x50
    cte3     equ 60h    ; Define una constante con valor 0x60

    org 0             ; Establece el origen en la dirección 0x0000
    goto inicio       ; Salta a la etiqueta "inicio"

    org 5             ; Establece la dirección de inicio del programa en 0x0005

inicio:
    bsf STATUS,5      ; Cambia al banco de registros 1
    BCF STATUS,6      ; Asegura que estamos en el banco correcto (banco 1)
    MOVLW H'0'        ; Carga el valor 0 en el registro W
    MOVWF TRISB       ; Configura todos los pines del puerto B como salida
    BCF STATUS,5      ; Regresa al banco 0
    clrf PORTB        ; Limpia el puerto B (pone todos los pines en 0)

loop2:
    incf PORTB        ; Incrementa el valor de PORTB en 1 (incrementa el valor de los bits en el puerto)
    call retardo      ; Llama a la funcion de retardo para hacer visible el cambio
    goto loop2        ; Vuelve al inicio del bucle, incrementando nuevamente PORTB

retardo:
    movlw cte1        ; Carga la constante cte1 en W
    movwf valor1      ; Guarda el valor en la variable valor1
tres:
    movlw cte2        ; Carga la constante cte2 en W
    movwf valor2      ; Guarda el valor en la variable valor2
dos:
    movlw cte3        ; Carga la constante cte3 en W
    movwf valor3      ; Guarda el valor en la variable valor3
uno:
     decfsz valor3     ; Decrementa valor3, si llega a 0, salta la siguiente instrucción
     goto uno          ; Si valor3 no es 0, repite el ciclo

     decfsz valor2     ; Decrementa valor2, si llega a 0, salta la siguiente instrucción
     goto dos          ; Si valor2 no es 0, repite el ciclo

     decfsz valor1     ; Decrementa valor1, si llega a 0, salta la siguiente instrucción
     goto tres         ; Si valor1 no es 0, repite el ciclo

     return

     END
```

Este programa configura el puerto B del PIC16F877 como salida y genera una secuencia en la que el valor del puerto B se incrementa continuamente. Como resultado, si se conectan LEDs al puerto B, se verá un efecto de conteo binario en el que los LEDs encenderán en un patrón de conteo ascendente. Se usa una subrutina de retardo para que los cambios sean visibles. La subrutina de retardo usa tres variables (valor1, valor2 y valor3) para generar una pausa en la ejecución mediante ciclos de decremento.

El código en ejecución es el siguiente:





En esta ejecución se puede ver el contador en binario se ejecuto de manera correcta a la esperada, los bits van recorriéndose como se solicitó, aunque es algo rápido, pero se podría reducir el tiempo para poder observarlo de mejor manera como se vio en la clase.

Hernández Diaz Sebastian

Conclusiones

En esta práctica aprendí a configurar y manipular el puerto B del PIC16F877 utilizando ensamblador, comprendí cómo inicializar los registros de control para definir los pines como entradas o salidas mediante el uso de TRISB, además de cómo limpiar el puerto con CLRF PORTB para evitar valores indeseados al inicio del programa, experimenté con la instrucción MOVLW para cargar valores en el registro W y con MOVWF para moverlos a registros específicos, entendí la importancia de BSF y BCF para modificar bits individuales del registro STATUS y poder cambiar entre bancos de memoria, aprendí a generar retardos utilizando ciclos anidados con DECFSZ que decrementa un valor almacenado en un registro y salta una instrucción cuando llega a cero, permitiendo así generar una pausa visible en la ejecución del programa, trabajé con la manipulación de bits en el puerto B aplicando MOVWF PORTB para actualizar el contenido del puerto y cambiar el estado de los pines, practiqué el uso de RRF y RLF que permiten desplazar bits hacia la derecha o izquierda respectivamente, logrando así implementar la secuencia de encendido de los LEDs, en la cual los bits de PORTB se desplazaban hasta completar la serie y reiniciarla, con el uso de INCF logré hacer un contador binario en el que el valor de PORTB aumentaba desde 00000000 hasta 11111111 generando un patrón de conteo visual en los LEDs conectados, comprendí la importancia de las estructuras de control como GOTO para mantener la ejecución en bucles continuos y CALL junto con RETURN para hacer llamadas a subrutinas de retardo optimizando el código y evitando repeticiones innecesarias, aprendí a manejar valores hexadecimales y binarios en ensamblador asegurando que la secuencia programada se reflejara correctamente en la salida, este conocimiento me permitió mejorar mi capacidad de programación en lenguaje ensamblador y fortalecer mi comprensión del funcionamiento de los microcontroladores en cuanto al control de puertos y manejo de retardos

Bibliografía

Microchip Technology Inc. (2001). *PIC16F87X Datasheet*. Recuperado de https://www.microchip.com

Peatman, J. B. (1998). *Embedded design with the PIC16F877 microcontroller*. Prentice Hall.

Mazidi, M. A., McKinlay, R. D., & Causey, D. (2008). *PIC microcontroller and embedded systems: Using assembly and C for PIC18*. Pearson Education.

Microchip Technology Inc. (2018). *PICmicro Mid-Range MCU Family Reference Manual*. Recuperado de https://ww1.microchip.com/downloads/en/DeviceDoc/33023a.pdf