

Objetivo.

Emplear los puertos paralelos que contiene un microcontrolador, para controlar la operación de dos motores de corriente directa, motores a pasos y servomotores.

Introducción

El manejo de motores mediante microcontroladores es un pilar fundamental en la robótica y la automatización, permitiendo el desarrollo de sistemas mecánicos controlados electrónicamente con precisión y eficiencia. Un microcontrolador, siendo un computador en miniatura en un solo circuito integrado, ofrece la capacidad de enviar señales programadas a motores de corriente continua (DC), servomotores, y motores paso a paso, facilitando así el control sobre la velocidad, dirección, y posición de estos dispositivos. A través de la configuración de puertos de entrada/salida y la implementación de algoritmos específicos, los desarrolladores pueden crear desde proyectos sencillos, como pequeños vehículos autónomos, hasta sistemas complejos como brazos robóticos y drones. La integración de sensores junto con técnicas avanzadas de programación permite además que estos sistemas reaccionen a su entorno, llevando a cabo tareas con un alto grado de autonomía y precisión.

Desarrollo de la práctica

Ejercicio 1

1.- Considerando la asignación de terminales asignadas en la figura 5.1; realizar el programa que ejecute el control indicado en la tabla

Nota: Las tierras de los ambos circuitos están conectados entre sí

La asignación de las terminales queda de la siguiente manera:

MOTOR2		
PC2	PB3	PB2
ENABLE M2	DIR1 M2	DIR2 M2

MOTOR1		
PC1	PB1	PB0
ENABLE M1	DIR1 M1	DIR2 M1

DATO Puerto Paralelo	ACCION	
	MOTOR M1	MOTOR M2
0x00	PARO	PARO
0x01	PARO	HORARIO
0x02	PARO	ANTI-HORARIO
0x03	HORARIO	PARO
0x04	ANTI-HORARIO	PARO
0x05	HORARIO	HORARIO
0x06	ANTI-HORARIO	ANTI-HORARIO
0x07	HORARIO	ANTI-HORARIO
0x08	ANTI-HORARIO	HORARIO

Tabla 5.1 Operación de motores de corriente directa

Propuesta de solución

Para el desarrollo de nuestra implementación utilizamos conocimientos adquiridos en las prácticas anteriores. Mover el motor de corriente directa fue sencillo gracias a la implementación del laboratorio simplemente hay que mandar las señales que necesitamos utilizando el modo de direccionamiento indexado para asegurarnos de que la salida mostrada en los motores corresponda con la entrada ingresada en PORTA

```
processor 16f877                ;Indica la versión de procesador
include <pl6f877.inc>          ;Incluye la librería de la versión del procesador

valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
cte1 equ 01h
cte2 equ 10h
cte3 equ 60h

ORG 0H                          ;Carga al vector de RESET la dirección de inicio
GOTO INICIO                     ;Nos movemos a la etiqueta inicio
ORG 05H                          ;Dirección de inicio del programa del usuario

INICIO:  CLRF    PORTA           ;se limpia el contenido del puerto PORTA
          BSF     STATUS,RP0     ;Cambia la banco 1
          BCF     STATUS,RP1
          MOVLW   06H            ;Configura puertos A y E como digitales
          MOVWF   ADCON1         ;
          MOVLW   3FH            ;Configura el puerto A como entrada
          MOVWF   TRISA          ;

          CLRF    TRISB          ;Se coloca un 0 en el registro W
          CLRF    TRISC          ; Se configura al puerto PORTB como puerto de salida
          BCF     STATUS,RP0     ; Se configura al puerto PORTC como puerto de salida
          ;Modo de direccionamiento indexado
          MOVF    PORTA,W        ; W <- (PORTA)
          ANDLW   0x0F           ;
          ADDWF   PCL,F          ;
          GOTO    CERO           ;PC + 0
          GOTO    UNO            ;PC * 1
          GOTO    DOS            ;PC * 2
          GOTO    TRES           ;PC + 3
          GOTO    CUATRO         ;PC + 4
          GOTO    CINCO          ;PC + 5
          GOTO    SEIS           ;PC + 6
          GOTO    SIETE          ;PC + 7
          GOTO    OCHO           ;PC + 8
          GOTO    CERO           ;PC + 9
          GOTO    CERO           ;PC + 10
          GOTO    CERO
          GOTO    CERO
          GOTO    CERO
          GOTO    CERO
          GOTO    CERO           ;PC + 15
```

CERO:	CALL	RETARDO
	MOVLW	B'00000110'
	MOVWF	PORTC
	CLRF	PORTB
	GOTO	LOOP
UNO:	CALL	RETARDO
	MOVLW	B'00000110'
	MOVWF	PORTC
	MOVLW	B'00000100'
	MOVWF	PORTB
	GOTO	LOOP
DOS:	CALL	RETARDO
	MOVLW	B'00000110'
	MOVWF	PORTC
	MOVLW	B'00001000'
	MOVWF	PORTB
	GOTO	LOOP
TRES:	CALL	RETARDO
	MOVLW	B'00000110'
	MOVWF	PORTC
	MOVLW	B'00000001'
	MOVWF	PORTB
	GOTO	LOOP
CUATRO:	CALL	RETARDO
	MOVLW	B'00000110'
	MOVWF	PORTC
	MOVLW	B'00000010'
	MOVWF	PORTB
	GOTO	LOOP
CINCO:	CALL	RETARDO
	MOVLW	B'00000110'
	MOVWF	PORTC
	MOVLW	B'00000101'
	MOVWF	PORTB
	GOTO	LOOP

SEIS:

```
CALL    RETARDO
MOVLW   B'00000110'
MOVWF   PORTC
MOVLW   B'00001010'
MOVWF   PORTB
GOTO    LOOP
```

SIETE:

```
CALL    RETARDO
MOVLW   B'00000110'
MOVWF   PORTC
MOVLW   B'00001001'
MOVWF   PORTB
GOTO    LOOP
```

OCHO:

```
CALL    RETARDO
MOVLW   B'00000110'
MOVWF   PORTC
MOVLW   B'00000110'
MOVWF   PORTB
GOTO    LOOP
```

RETARDO:

```
                MOVLW cte1
                MOVWF valor1
tres MOVLW cte2
                MOVWF valor2
dos  MOVLW cte3
                MOVWF valor3
uno  DECFSZ valor3
                GOTO uno
                DECFSZ valor2
                GOTO dos
                DECFSZ valor1
                GOTO tres
```

return

END ;Fin de programa

Ensamblado correctamente

```
Release build of project 'C:\Users\Alexis\Desktop\FI\Microcomputadoras\Lab\P5\e1.disposable_mcp' started.
Language tool versions: MPASMWIN.exe v5.51, mplink.exe v4.49, mplib.exe v4.49
Sun Apr 07 01:33:03 2024

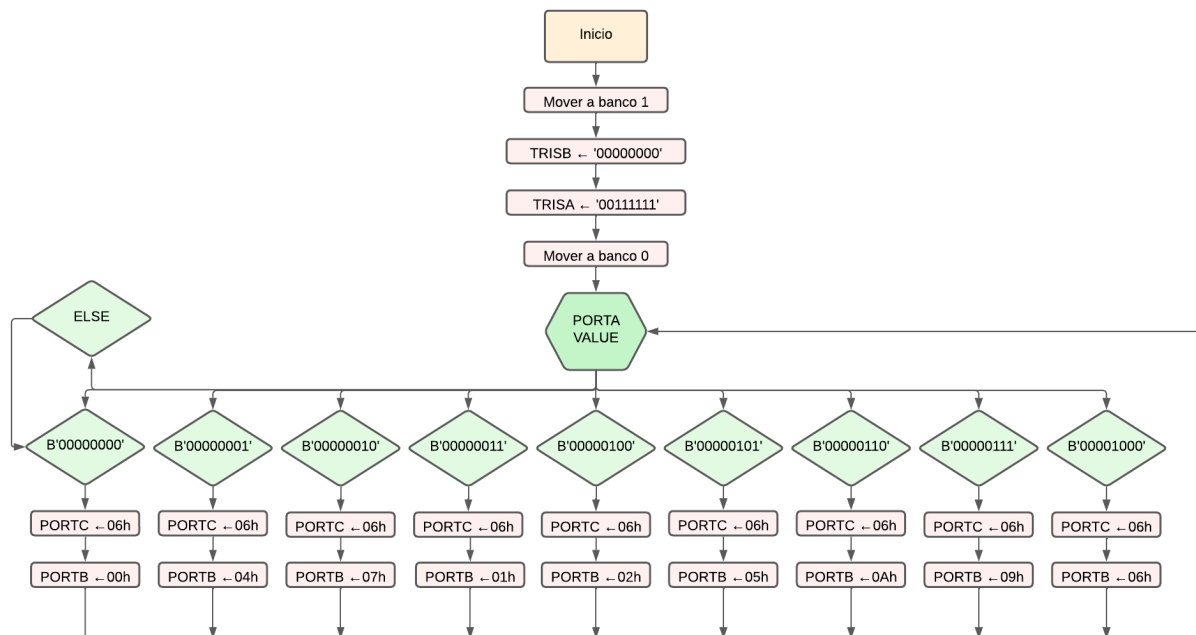
Clean: Deleting intermediary and output files.
Clean: Done.
Executing: "C:\Program Files (x86)\Microchip\MPASM Suite\MPASMWIN.exe" /q /p16F877A "e1.asm" /l"e1.lst" /e"e1.err"
Warning[205] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 1 : Found directive in column 1. (processor)
Warning[205] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 2 : Found directive in column 1. (include)
Message[301] C:\PROGRAM FILES (x86)\MICROCHIP\MPASM SUITE\P16F877.INC 33 : MESSAGE: (Processor-header file mismatch. Verify selected processor.)
Warning[207] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 3 : Found label after column 1. (valor1)
Message[302] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 17 : Register in operand not in bank 0. Ensure that bank bits are correct.
Message[302] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 19 : Register in operand not in bank 0. Ensure that bank bits are correct.
Message[302] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 21 : Register in operand not in bank 0. Ensure that bank bits are correct.
Message[302] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 22 : Register in operand not in bank 0. Ensure that bank bits are correct.
Message[305] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 115 : Using default destination of 1 (file).
Message[305] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 117 : Using default destination of 1 (file).
Message[305] C:\USERS\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E1.ASM 119 : Using default destination of 1 (file).
Executing: "C:\Program Files (x86)\Microchip\MPASM Suite\mplink.exe" /p16F877A "e1.o" /z_MPLAB_BUILD=1 /o"e1.cof" /M"e1.map" /W/x
MPLINK 4.49, Linker
Device Database Version 1.14
Copyright (c) 1998-2011 Microchip Technology Inc.
Errors : 0

Loaded C:\Users\Alexis\Desktop\FI\Microcomputadoras\Lab\P5\e1.cof.

Release build of project 'C:\Users\Alexis\Desktop\FI\Microcomputadoras\Lab\P5\e1.disposable_mcp' succeeded.
Language tool versions: MPASMWIN.exe v5.51, mplink.exe v4.49, mplib.exe v4.49
Sun Apr 07 01:33:04 2024

BUILD SUCCEEDED
```

Diagrama de flujo del código:



Ejercicio 2

2.-Realizar un programa que controle la cantidad de pasos que debe dar un motor, así como el sentido de giro

Dato Puerto Paralelo	Motor a pasos
0x00	Motor en paro
0x01	Gira en sentido horario
0x02	Gira en sentido anti horario
0x03	Gira cinco vueltas en sentido horario
0x04	Gira 10 vueltas en sentido anti horario

Tabla 5.2 Control del motor a pasos

Propuesta de solución

En este ejercicio requeríamos hacer rotaciones hacia la izquierda o derecha según el sentido de giro del motor a pasos, para ahorrarnos comparaciones se realizo asignaciones con direccionamiento directo al registro PORTB en lugar de rotaciones y para controlar el número de giros en lugar de mantenernos en un LOOP infinito utilizamos la instrucción SLEEP para detener el flujo y poder contabilizar los

```
processor 16f877
include<p16f877.inc>
;VALORES Para la rutina de retardo
valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
cte1 equ 11h
cte2 equ 50h
cte3 equ 60h
org 0h
goto INICIO
org 05h
;VALORES PARA RETRASOS DE CADA ESTADO
contadorEdo1 equ 0x24
contadorEdo2 equ 0x25

INICIO:

    clrf PORTA
    bsf STATUS,RP0 ;Cambia la banco 1
    bcf STATUS,RP1
    movlw h'00'
    movwf TRISB ;Configura puerto B como salida
    clrf PORTB
    movlw 06h ;Configura puertos A y E como digitales
    movwf ADCON1
    movlw 3fh ;Configura el puerto A como entrada
    movwf TRISA
    bcf STATUS,RP0 ;regresa al banco 0

LOOP:

    MOVF    PORTA,W      ; W <- (PORTA)
    ANDLW   0x07         ;
    ADDWF   PCL,F        ;
    GOTO    CERO          ;PC + 0
    GOTO    UNO           ;PC * 1
    GOTO    DOS           ;PC * 2
    GOTO    TRES          ;PC + 3
    GOTO    CUATRO        ;PC + 4
    GOTO    CERO          ;PC + 5
    GOTO    CERO          ;PC + 6
    GOTO    CERO          ;PC + 7
```

giros del motor.

```
CERO:
    MOVLW 0X00
    MOVWF PORTB
    GOTO LOOP

UNO:
    MOVLW 0X01
    MOVWF contadorEdo1

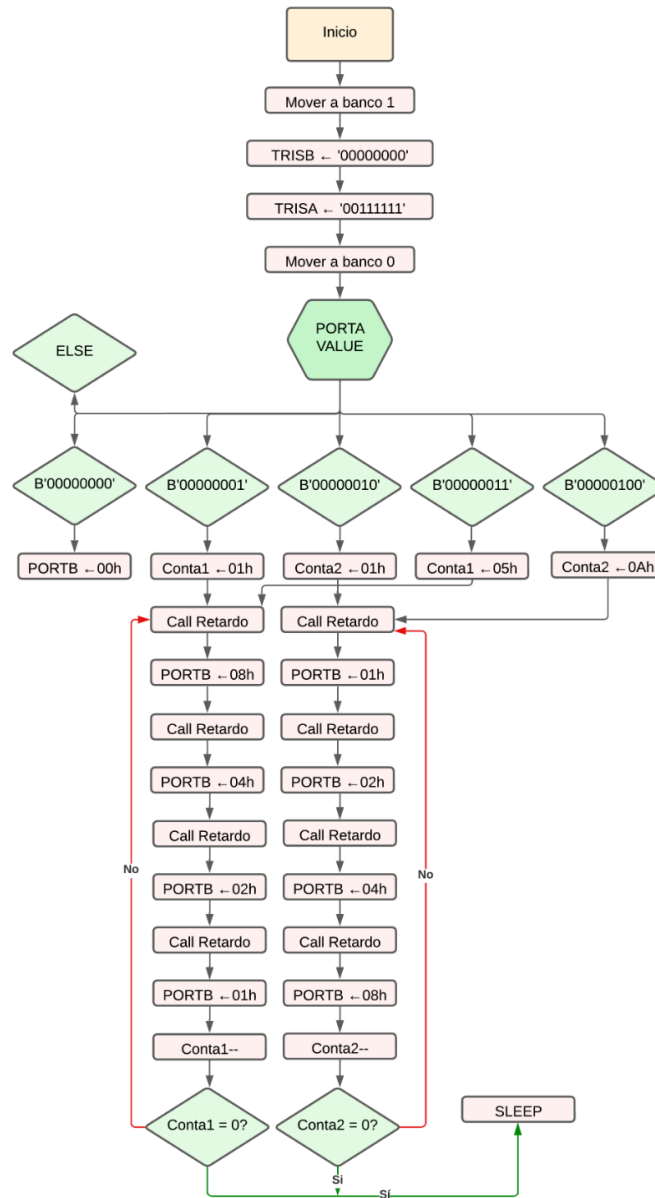
LOOP_ESTADO_1
    CALL retardo
    MOVLW b'00001000'
    MOVWF PORTB
    CALL retardo
    MOVLW b'00000100'
    MOVWF PORTB
    CALL retardo
    MOVLW b'00000010'
    MOVWF PORTB
    CALL retardo
    MOVLW b'00000001'
    MOVWF PORTB
    CALL retardo
    DECFSZ contadorEdo1
    GOTO LOOP_ESTADO_1
    SLEEP

DOS:
    MOVLW 0X01
    MOVWF contadorEdo2

LOOP_ESTADO_2
    CALL retardo
    MOVLW b'00000001'
    MOVWF PORTB
    CALL retardo
    MOVLW b'00000010'
    MOVWF PORTB
    CALL retardo
    MOVLW b'00000100'
    MOVWF PORTB
    CALL retardo
    MOVLW b'00001000'
    MOVWF PORTB
    CALL retardo
    DECFSZ contadorEdo2
    GOTO LOOP_ESTADO_2
    SLEEP
```

	<pre> SLEEP TRES: MOVLW 0X05; MOVWF contadorEdo1 GOTO LOOP_ESTADO_1 CUATRO: MOVLW 0X0A MOVWF contadorEdo2 GOTO LOOP_ESTADO_2 retardo movlw cte1 ;Rutina que genera un DELAY movwf valor1 tres movwf cte2 movwf valor2 dos movlw cte3 movwf valor3 uno decfsz valor3 goto uno decfsz valor2 goto dos decfsz valor1 goto tres return end </pre>
<p>Ensamblado correctamente</p>	<pre> Release build of project 'C:\Users\Alexis\Desktop\FI\Microcomputadoras\Lab\P5\e2\disposable_mcp' started. Language tool versions: MPASMWIN.exe v5.51, mplink.exe v4.49, mplib.exe v4.49 Sun Apr 07 02:25:54 2024 Clean: Deleting intermediary and output files. Clean: Done. Executing: "C:\Program Files (x86)\Microchip\MPASM Suite\MPASMWIN.exe" /q /p16F877A "e2.asm" /I"e2.lst" /e"e2.err" Warning[205] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 1: Found directive in column 1. (processor) Warning[205] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 2: Found directive in column 1. (include) Message[301] C:\PROGRAM FILES (X86)\MICROCHIP\MPASM SUITE\P16F877.INC 33: MESSAGE: (Processor-header file mismatch. Verify selected processor.) Warning[205] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 10: Found directive in column 1. (org) Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 11: Found opcode in column 1. (goto) Warning[205] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 12: Found directive in column 1. (org) Message[302] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 22: Register in operand not in bank 0. Ensure that bank bits are correct. Warning[302] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 25: Register in operand not in bank 0. Ensure that bank bits are correct. Message[302] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 27: Register in operand not in bank 0. Ensure that bank bits are correct. Message[305] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 64: Using default destination of 1 (file). Message[305] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 84: Using default destination of 1 (file). Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 97: Found opcode in column 1. (movwf) Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 99: Found opcode in column 1. (movwf) Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 101: Found opcode in column 1. (movwf) Message[305] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 102: Using default destination of 1 (file). Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 103: Found opcode in column 1. (goto) Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 104: Found opcode in column 1. (decfsz) Message[305] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 104: Using default destination of 1 (file). Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 105: Found opcode in column 1. (goto) Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 106: Found opcode in column 1. (decfsz) Message[305] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 106: Using default destination of 1 (file). Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 107: Found opcode in column 1. (goto) Warning[203] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 108: Found opcode in column 1. (return) Warning[205] C:\Users\ALEXIS\DESKTOP\FI\MICROCOMPUTADORAS\LAB\P5\E2.ASM 109: Found directive in column 1. (end) Executing: "C:\Program Files (x86)\Microchip\MPASM Suite\mplink.exe" /p16F877A "e2.o" /z _MPLAB_BUILD=1 /o"e2.cof" /M"e2.map" /W /X MPLINK V4.49, Linker Device Database Version 1.14 Copyright (c) 1998-2011 Microchip Technology Inc. Errors : 0 Loaded C:\Users\Alexis\Desktop\FI\Microcomputadoras\Lab\P5\e2.cof. Release build of project 'C:\Users\Alexis\Desktop\FI\Microcomputadoras\Lab\P5\e2\disposable_mcp' succeeded. Language tool versions: MPASMWIN.exe v5.51, mplink.exe v4.49, mplib.exe v4.49 Sun Apr 07 02:25:55 2024 BUILD SUCCEEDED </pre>

Diagrama de flujo del código:



Ejercicio 3

3.- Utilizando un servo motor realizar el control mostrado en la tabla No. 5.3




SW2	SW1	SW0	Posición Servo	Representación
1	0	0	Izquierda	 0°
0	1	0	Central	 90°
0	0	1	Derecha	 180°

Tabla 5.3 Funcionamiento del servo motor

<p>Propuesta de solución</p> <p>La dificultad de este ejercicio consistió en comprender la complejidad con la que funcionan los retardos en ensamblador cuando requerimos de tiempos precisos, en esta caso, para simular un PWM de 3 posiciones requerimos pulsos de 1, 1.5 y 2 milisegundos para representar correctamente los grados del servomotor por lo que al final resultó ser un ejercicio de prueba y error para encontrarnos con los valores más óptimos para cumplir la tarea que esperábamos. Calcular el valor de los ciclos de instrucción en nuestro</p>	<pre> processor 16f877 include<p16f877.inc> CTE1 EQU 0X20 CTE2 EQU 0X21 ; Variables adicionales en el área CBLOCK si son necesarias MILIS EQU 0X22 ; Contador para los milisegundos MICROS EQU 0X23 ; Contador para los microsegundos adicionales en retardo de 1.5ms CUENTA EQU 0X24 ORG 0 GOTO INICIO ORG 5 INICIO: CLRF PORTA ;se limpia el contenido del puerto PORTA BSF STATUS,RP0 ;Cambia la banco 1 BCF STATUS,RP1 MOVLW 06H ;Configura puertos A y E como digitales MOVWF ADCON1 ; MOVLW 3FH ;Configura el puerto A como entrada MOVWF TRISA CLRF TRISB ;Se coloca un 0 en el registro W BCF STATUS,RP0 ; Se configura al puerto PORTB como puerto de salida ;Modo de direccionamiento indexado LOOP: MOVF PORTA,W ; W <- (PORTA) ANDLW 0x07 ; ADDWF PCL,F ; GOTO CERO ;PC + 0 GOTO UNO ;PC * 1 GOTO DOS ;PC * 2 GOTO TRES ;PC + 3 GOTO CERO ;PC + 4 GOTO CERO ;PC + 5 GOTO CERO ;PC + 6 GOTO CERO ;PC + 7 </pre>
---	--

controlador de 20Mhz resultó en un tiempo de 200 nS por ciclo de instrucción, valor que nos ayudó a contabilizar los ciclos y aproximarnos a los resultados sin la necesidad de navegar sin ningún tipo de guía en esta aproximación de valores.

CERO:

```
CLRF    PORTB
GOTO    LOOP
```

UNO:

```
MOVLW   B'00000001'
MOVWF   PORTB
MOVLW   D'5'
MOVWF   CTE2
CALL    RETARDO
CLRF    PORTB
MOVLW   D'100'
MOVWF   CTE2
CALL    RETARDO
GOTO    LOOP
```

DOS:

```
MOVLW   B'00000001'
MOVWF   PORTB
MOVLW   D'7'
MOVWF   CTE2
CALL    RETARDO
CLRF    PORTB
MOVLW   D'100'
MOVWF   CTE2
CALL    RETARDO
GOTO    LOOP
```

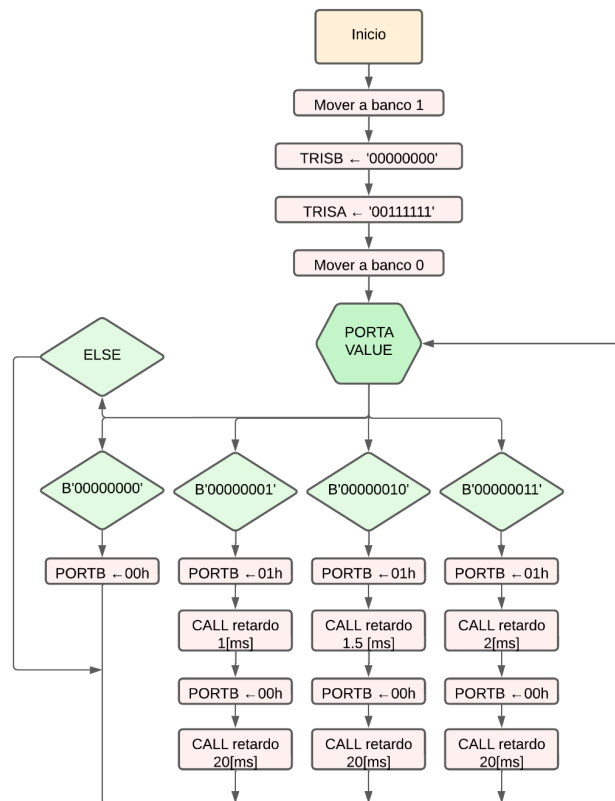
TRES:

```
MOVLW   B'00000001'
MOVWF   PORTB
MOVLW   D'10'
MOVWF   CTE2
CALL    RETARDO
CLRF    PORTB
MOVLW   D'100'
MOVWF   CTE2
CALL    RETARDO
GOTO    LOOP
```



	<div><div>RETARDO:</div><div><div>BUCLE2:</div><div>MOVLW</div><div>D'250'</div><div>;1cy</div></div><div><div>MOVWF</div><div>CTE1</div><div>;1cy</div></div><div>BUCLE1:</div><div><div>NOP</div><div>;1cy</div></div><div><div>DECFSZ</div><div>CTE1,F</div><div>;1cy</div></div><div><div>GOTO</div><div>BUCLE1</div></div><div><div>DECFSZ</div><div>CTE2,F</div></div><div><div>GOTO</div><div>BUCLE2</div></div><div>RETURN</div></div> <div>END</div>
--	---

Diagrama de flujo del código:



Simulación

<https://youtu.be/oHxr0ocLjw8>

Análisis técnico

1. ¿Se puede comprobar que la solución producida funciona?

Gracias a la implementación de circuitos físicos pudimos observar el correcto funcionamiento de los ejercicios realizados durante la clase, además, para las actividades restantes se pudo verificar el funcionamiento a través del uso de un circuito simulado.

2. ¿Se alcanzó el objetivo?

Sí, logramos implementar una solución efectiva para cada actividad. Además, pudimos verificar el correcto desempeño de nuestros códigos gracias a los circuitos físicos como simulados.

Ejercicio 1

¿Cuál es el flujo interno de los datos?

El flujo interno de los datos comienza con la lectura del puerto PORTA en el bucle principal LOOP. Se aplica una máscara (ANDLW 0x0F) para mantener solo los 4 bits menos significativos. Luego, se utiliza un salto condicional basado en el valor resultante para dirigir el flujo del programa a diferentes partes del código, determinado por las etiquetas CERO, UNO, DOS, TRES, CUATRO, CINCO, SEIS, SIETE y OCHO.

¿Cuáles fueron los modos de direccionamiento utilizados?

Durante el ejercicio hicimos uso del direccionamiento indexado en el bucle principal LOOP, donde se realiza un salto a una dirección a partir del valor de PORTA y el contador de programa mediante la instrucción ADDWF PCL,F, lo que permite ejecutar diferentes segmentos de código según el valor de PORTA.

Así mismo, usamos el direccionamiento directo en las instrucciones GOTO dentro del bucle principal LOOP, para saltar a las etiquetas CERO, UNO, DOS, TRES, CUATRO, CINCO, SEIS, SIETE y OCHO, dependiendo del valor de PORTA.

Ejercicio 2

¿Cuál es el flujo interno de los datos?

Al igual que en el ejercicio anterior el flujo interno de datos comienza con la lectura del puerto PORTA en el LOOP. Se aplica una máscara (ANDLW 0x07) para mantener solo los 3 bits menos significativos y se utiliza un salto condicional para dirigir el flujo del programa a diferentes partes del código, determinado por las etiquetas CERO, UNO, DOS, TRES y CUATRO.

¿Cuáles fueron los modos de direccionamiento utilizados?

Se usaron los modos de direccionamiento indexado en el LOOP, donde se realiza un salto a una dirección a partir del valor de PORTA y el contador de programa (PCL). Mediante la instrucción ADDWF PCL,F, lo que permite ejecutar diferentes segmentos de código según el valor de PORTA.

También se utiliza el modo de direccionamiento directo en las instrucciones GOTO dentro del bucle principal LOOP, para saltar a las etiquetas CERO, UNO, DOS, TRES y CUATRO, dependiendo del valor de PORTA.

Y finalmente dentro de cada etiqueta de salto, se utilizan principalmente los modos de direccionamiento inmediato y de registro, donde se cargan constantes en registros específicos (MOVLW) y se escriben en puertos específicos (MOVWF)..

Ejercicio 3

¿Cuál es el flujo interno de los datos?

El flujo interno de datos en este código se mueve desde el bucle principal LOOP, se lee el estado del puerto PORTA, y se realiza una máscara (ANDLW 0x07) para mantener solo los 3 bits menos significativos. Luego, dependiendo del valor obtenido, se realiza un salto condicional a una de las etiquetas CERO, UNO, DOS o TRES, donde se ejecuta cierto código según el valor del puerto PORTA.

Dentro de cada etiqueta de salto, se ejecutan diferentes instrucciones según el valor del puerto PORTA, lo que afecta el estado del puerto PORTB y luego se vuelve al bucle principal LOOP

¿Cuáles fueron los modos de direccionamiento utilizados?

Los modos de direccionamiento utilizados fueron dos, el primero, el modo de direccionamiento indexado en el bucle principal LOOP, donde se realiza un salto a una dirección a partir del valor de PORTA y el contador de programa (PCL), permitiendo así ejecutar diferentes segmentos de código basados en el valor de PORTA. Y el segundo, el modo de direccionamiento directo en las instrucciones GOTO dentro del bucle principal LOOP, para saltar a las etiquetas CERO, UNO, DOS o TRES.

Conclusiones

Alcantar Correa Vianey:

En esta práctica consolidé mi entendimiento sobre la configuración de puertos de entrada y salida en el PIC16F877, aplicando este conocimiento al control de motores DC, paso a paso y servomotores a través de un nuevo puerto de salida en PORT C y monitoreando entradas en PORT A mediante switches. Este enfoque me permitió navegar entre distintos escenarios operativos para los motores, enfatizando la importancia de revisar las asignaciones de la tarjeta antes de programar. Este aprendizaje no solo refuerza mi capacidad para configurar puertos según las necesidades del proyecto, sino que también abre posibilidades para desarrollar proyectos más complejos que involucren distintos tipos de motores.

Sanchez Rosas Alexis Alejandro:

El desarrollo de esta práctica nos ayuda a entender como trabaja en microcontrolador con algunos dispositivos como los motores de diferentes tipos, en nuestra formación a lo largo de la carrera habíamos trabajado con estos tipos de motores por lo que la práctica nos ayudó a adicionar conocimiento en cuanto a las posibilidades y facilidades que representa el uso de un microcontrolador para el control de los motores. El código de las prácticas a primera vista era sencillo pero nos representó algunos problemas en cada ejercicio, finalmente conseguimos cumplir con lo solicitado en tiempo y en forma gracias al trabajo en equipo.

Velazquez Martinez Karla Andrea:

Durante la realización de esta práctica, exploramos la interacción de nuestro controlador con motores, lo cual resultó sumamente interesante . Utilizamos los distintos puertos y rutinas de retardo para establecer comunicación con cada motor de manera individual, aprovechando el puerto A como entrada y los puertos B y C como salidas respectivamente.

Con cada práctica vamos obteniendo mayor conocimiento sobre el microcontrolador, descubriendo nuevas técnicas y aplicaciones . Y esto nos permite resolver las actividades de cada ejercicio respectivamente aunque tuvimos algunas complicaciones logramos resolverlas.

Bibliografía

Microchip Technology Inc. (s/f). PIC16F87X Data Sheet. Recuperado de <https://ww1.microchip.com/downloads/en/DeviceDoc/31029a.pdf>

S.a.. (1997). Section 29. Instruction Set - Microchip Technology. USA:
S.e..DESCRIPCIÓN DELAS INSTRUCCIONES. (s. f.). Profesores Sanvalero