UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Práctica 3: Sistema mínimo microcontrolador PIC16F877

INTEGRANTES DE EQUIPO Y GRUPO DE TEORÍA

López Becerra Ricardo - 420053710 - GRUPO 3
Navarrete Zamora Aldo Yael - 317242409 - GRUPO 4

ASIGNATURA

Laboratorio de Microcomputadoras

GRUPO DE LABORATORIO

8

FECHA DE REALIZACIÓN

23 de febrero del 2023

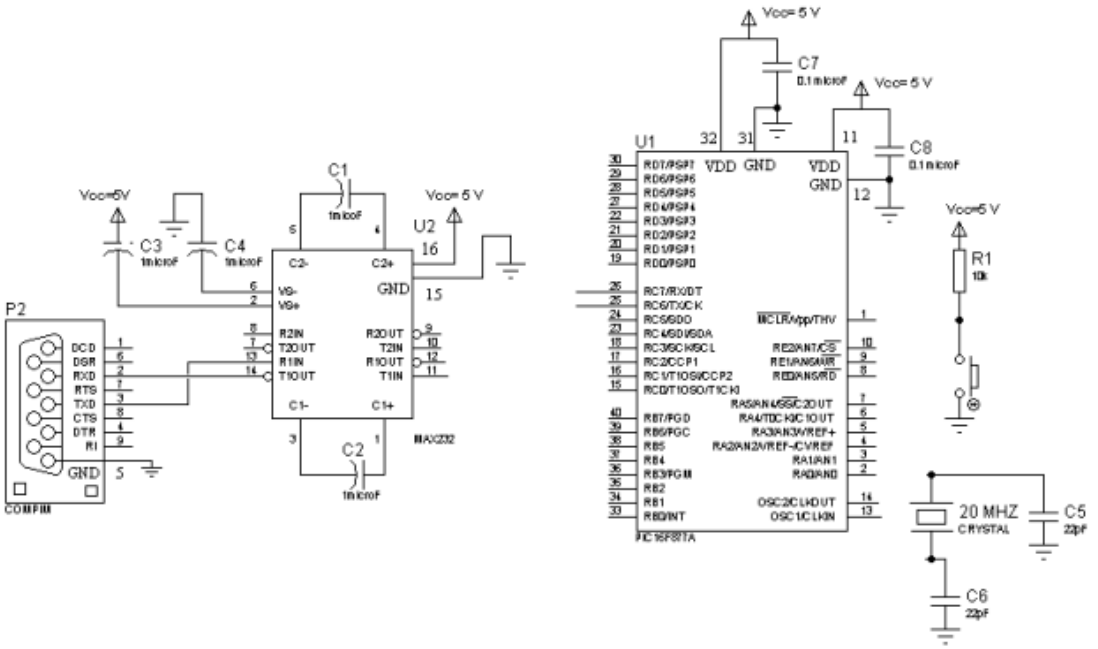
FECHA DE ENTREGA

16 de marzo del 2023

1. Desarrollo

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

El sistema utilizado para esta práctica está diseñado de acuerdo al siguiente diagrama:



1. Revisar a detalle y en concordancia con el circuito 3.2, identificar las conexiones faltantes, discutir con sus compañeros y con su profesor(a) el impacto y función de los mismos.

   Las conexiones faltantes son las siguientes:

   a) La conexión entre el microprocesador y el MAX232. La falta de esta conexión haría imposible cargar programas en el microprocesador sin necesitas de un programador externo.

   b) el circuito de reset: Evitaría que pudiéramos reiniciar la ejecución del código en el microprocesador, por lo que tampoco podríamos cargar nuevos programas sin desconectar de la corriente al microcontrolador.

   c) La conexión con el reloj: El microcontrolador no funcionaría ya que no tendría como cordinar sus operaciones.

2. Completar las conexiones faltantes, utilizando jumpers; cerciorar el alambrado correcto.

3. Una vez resueltos las actividades anteriores, identificar la terminal PB0 del puerto B, realizar la conexión con la salida de una resistencia y un led.

   Los circuitos se nos dieron ya conectados, por lo que esto no fue necesario realizar los dos puntos anteriores.

4. Escribir, comentar e indicar que hace el siguiente programa.

Código comentado

```
processor 16f877
include <p16f877.inc>
valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
cte1 equ 20h
cte2 equ 50h
cte3 equ 60h

    ORG 0
    GOTO INICIO
    ORG 5
INICIO:
;Selecciona el banco
;de memoria 1
    BSF STATUS,RP0
    BCF STATUS,RP1
;Configura todos los
;bits de puerto como
;salida
    MOVLW H'0'
    MOVWF TRISB
; Regresa al banco
;de memoria 0
    BCF STATUS,RP0
;Limpia todos los
;bits del puerto B
    CLRF PORTB
loop2
;Enciende el led 0
;encendiendo el bit
;0 del puerto B
    BSF PORTB,0
;LLamada a rutina de
;retardo
    CALL retardo
;Apaga el led 0
;apagando el bit 0 del
;puerto B
    BCF PORTB,0
    CALL retardo
    GOTO loop2

;Rutina de retardo con
;varios ciclos anidados
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
```

Este código lo que hace es prender y apagar uno de los leds de la tarjeta. Esto lo realiza limpiando y borrando el contenido del bit 0 del puerto B del microprocesador en un loop infinito y realizando un retraso entre las dos operaciones.

Algoritmo

El algoritmo es muy simple. En un loop infinito, se prende el bit 0 del puerto B para prender el led. Después, se llama a la rutina de retardo, que consiste de tres loops anidados. Finalmente se pone apaga el bit 0 del puerto B para apagar el led y se llama a la subrutina retardo una vez más.

5. Ensamblar y cargar el programa anterior en el microcontrolador ¿Qué es lo que puede visualizar?

   Se visualiza como el led apaga y prende con un ritmo controlado.

6. - En el programa, modifique el valor de cte1 a 8h, ensamblar y programar ¿Qué sucede y por qué?

El led comienza a parpadear más rápido ya que los ciclos de la subrutina de retardo duran significativamente menos.

7. Modifique cte1 a 80h; ensamblar y programar ¿Existe algún cambió?

   El Ritmo de parpadeo ahora es más lento.

8. 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.

   Para realizar este punto se modifico ligeramente el código del ejercicio anterior.

Algoritmo

El algoritmo es casi el mismo que en el ejercicio anterior. La única diferencia es que en lugar de prender y apagar únicamente el bit menos significativo del puerto B ahora se prenden y apagan todos. Para encenderlos todos se carga la constante 0xFF en el registro PORTB y para apagar todos los bits simplemente se hace uso de la instrucción CLRF.

Código comentado

```
processor 16f877
include <p16f877.inc>
valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
cte1 equ 20h
cte2 equ 50h
cte3 equ 60h
        ORG 0
GOTO INICIO
        ORG 5
INICIO:
;Configura el puerto B
;como salida
        BSF STATUS,RP0
        BCF STATUS,RP1
        MOVLW H'0'
        MOVWF TRISB
        BCF STATUS,RP0
        CLRF PORTB
loop2
;Carga la constante FF
        MOVLW H'FF'
        MOVWF PORTB
        CALL retardo
;Limpia los bits de
;registro
        CLRF PORTB
        CALL retardo
        GOTO loop2
;Un retardo
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
```

9. Realizar un programa que muestre la siguiente secuencia en el puerto B con retardos de 1/2 segundo.

   Este programa al igual que los anteriores cambia ligeramente, seguimos manteniendo el puerto B como salida.

Algoritmo

La diferencia para este enunciado es que inicialmente ponemos el valor del PORTB H'80', es decir, el bit más significativo estará encendido, y dentro del loop, se realiza un retardo con el tiempo especificado.

Finalmente, el ciclo seguirá iterando, y cuando haya un desbordamiento automáticamente el PIC regresará el led al lugar original.

Código comentado

```
processor 16f877
include <p16f877.inc>
; Registros para guardar los
;contadores auxiliares para
;el retardo.
valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
; Constantes que definen
; la duracion del retardo.
cte1 equ 20h
cte2 equ 50h
cte3 equ 60h
        ORG 0
GOTO INICIO
        ORG 5
INICIO:
    ;Nos posicionamos en el
    ;banco 01
        BSF STATUS,RP0
        BCF STATUS,RP1
    ;Configuramos el TRISB para
    ;que sea 'SALIDA'
        MOVLW H'00'
        MOVWF TRISB
    ;Direccionamiento directo
        BCF STATUS,RP0
    ;Ponemos como valor inicial
    ;B'10000000' en el PORTB
        MOVLW B'10000000'
        MOVWF PORTB
loop2
        CALL retardo
    ;Hacemos el corrimiento
    ;a la derecha de PORTB
        RRF PORTB
        GOTO loop2
retardo ;INICIA EL 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
```

10. Realizar un programa que controle el funcionamiento de dos semáforos, cada estado tendrá una duración de 2 segundos.

Algoritmo

En un loop infinito se definen los cuatro estados del semáforo. para cada estado primero se limpia el contenido del registro PORTB, luego se prenden los bits correspondientes a los leds que es necesario encender en el estado con la instrucción BSF. Finalmente, se llama a la subrutina retardo.

Código comentado

```
processor 16f877
include <p16f877.inc>
valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
;Aumento de la constante 1
; Para aumentar el tiempo
; entre parpadeos
cte1 equ h'ff'
cte2 equ 50h
cte3 equ 60h
        ORG 0
GOTO INICIO
        ORG 5
INICIO:
;Programa el puerto B
;como salida
        BSF STATUS,RP0
        BCF STATUS,RP1
        MOVLW H'00'
        MOVWF TRISB
        BCF STATUS,RP0
        MOVLW B'00000000'
        MOVWF PORTB
loop2
; Se definen cada uno de
; los estados del semaforo.
; Se limpian todos
; los bits del puerto B
; se prender los bits
; necesarios y se llama
; a la subrutina de retraso

; ESTADO 1
        CLRF PORTB
        BSF PORTB, 6
        BSF PORTB, 0
        CALL retardo
; ESTADO 2
        CLRF PORTB
        BSF PORTB, 5
        BSF PORTB, 0
        CALL retardo
; ESTADO 3
        CLRF PORTB
        BSF PORTB, 4
        BSF PORTB, 2
        CALL retardo
; ESTADO 4
        CLRF PORTB
        BSF PORTB, 4
        BSF PORTB, 1
        CALL retardo
        GOTO loop2

;Surutina de retardo
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
```

2. Conclusiones

- López Becerra Ricardo: Con los ejercicios realizados aprendimos cual es el sistema mínimo con el que funciona el microprocesador pic16 y cual es la manera de interactuar con el hardware a través de los puertos paralelos. Para realizar esto aprendimos sobre los registros TRISTX y PORTB. El primero para indicar cuales bits del puerto son de entrada y cuales de salida y el segundo para escribir los datos en la salida. Finalmente, para hacer los efectos visibles a los humanos, aprendimos a implementar una subrutina que nos permite gastar tiempo a través de varios loops anidados.
- Navarrete Zamora Aldo Yael: Esta práctica fue de mucha ayuda para realizar las manipulaciones de los puertos, en ese caso la manipulación del PORTB como puerto de salida no fue trivial, pero tampoco fue tan complicado, esto nos permite entender a estos puertos paralelos. Espero que más adelante podamos configurar el TRISTX para el puerto paralelo A, y podamos recibir alguna señal de entrada en dichos puertos.