M16

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Práctica 6:

Convertidor Analógico/Digital

Profesor:

M.I. Ruben Anaya García

Alumnos:	Número de cuenta:
Barreiro Valdez Alejandro	317520888
Zepeda Baeza Jessica	317520747

Grupo Laboratorio: 4

Grupo Teoría: 1

Fecha de realización: 25 de abril de 2023

Fecha de entrega: 29 de abril de 2023

Desarrollo

Para esta práctica se buscó crear programas en ensamblador que hicieran uso del convertidor analógico digital del microcontrolador PIC16F877. Para ello, lo primero que se realizó fue entender la manera en que funciona este convertidor. Se tienen 8 canales de entrada que procesan señales analógicas transformando la información en 10 bits. También se revisó el algoritmo para hacer uso del convertidor. Entre los pasos más relevantes están: convertir el puerto en analógico utilizando el registro ADCON1; configurar la fuente de reloj y el canal de entrada y prender el convertidor con el registro ADCON0; iniciar y esperar la conversión; y obtener el resultado con ADRESH. Con estos conocimientos se tuvo una idea de qué es lo que se buscaba desarrollar utilizando el convertidor analógico/digital.

Los programas desarrollados fueron:

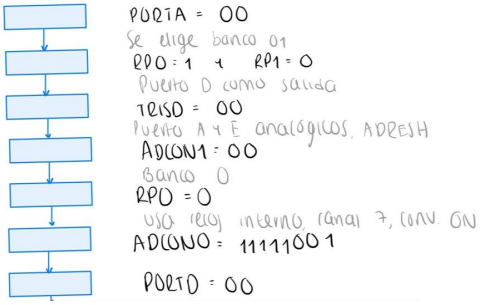
1. Un programa donde de acuerdo a una entrada analógica generada con un potenciómetro, se representa la conversión utilizando LEDs y un convertidor de 7 segmentos.
2. Un programa que indica el rango en donde se encuentra el voltaje de entrada utilizando un display de 7 segmentos y los siguientes rangos:

Entrada Analógica Ve	Salida
0 – 0.99 V	0
1.0 – 1.99 V	1
2.0 – 2.99 V	2
3.0 – 3.99 V	3
4.00 – 4.80 V	4
4.80 – 5.00 V	5

3. Un programa que identifica tres señales analógicas y las compara para obtener cuál es la mayor de las tres. El resultado de salida será 1, 3 o 7 en el display de 7 segmentos dependiendo de cuál es el mayor.

Algoritmos

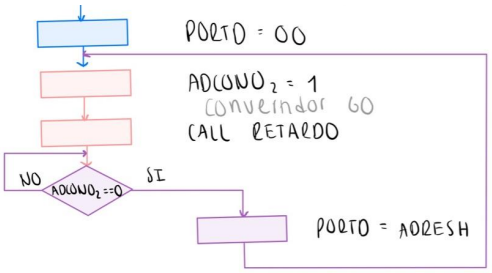
La configuración inicial del puerto de entrada E y el puerto de salida D se muestra en la siguiente carta ASM:



En esta configuración se elige el convertidor del canal 7. Esta configuración se utiliza al inicio de los primeros dos ejercicios.

Ejercicio 1

La carta ASM para el Ejercicio 1 es la siguiente:



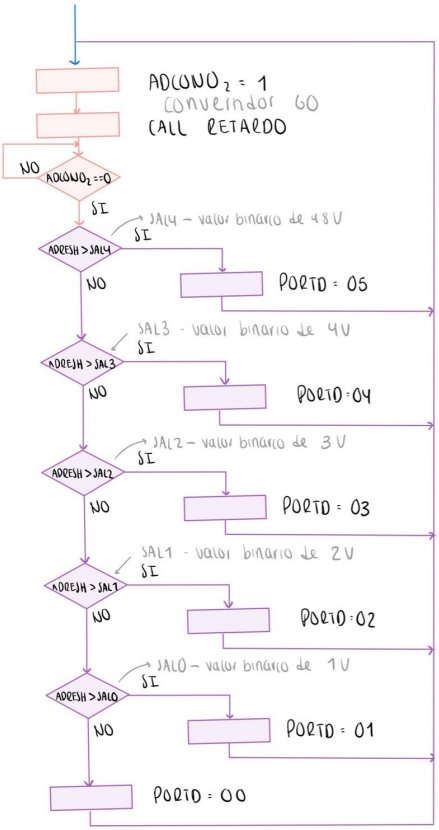
Lo que hace el programa es, una vez elegido el canal y encendido el convertidor, prende la bandera de GO/DONE para indicarle al convertidor que puede iniciar. Después llama un pequeño retardo para esperar el proceso de conversión y una vez que acaba, se verifica si la bandera GO/DONE es cero. Si es así significa que ya acabó de convertir entonces el valor ubicado en *ADRESH* se pasa al puerto de salida D para observarse en el Display de 7 segmentos y en los LEDs. Si aún no acaba de convertir, se vuelve a verificar la bandera hasta que sea cero. Una vez que se muestra el dato de salida, se vuelve a prender la bandera de GO para realizar otra conversión.

Ejercicio 2

Para este ejercicio primero se anotaron los valores binarios que tienen los voltajes de 4.8, 4, 3, 2 y 1. Se obtuvieron los siguientes valores:

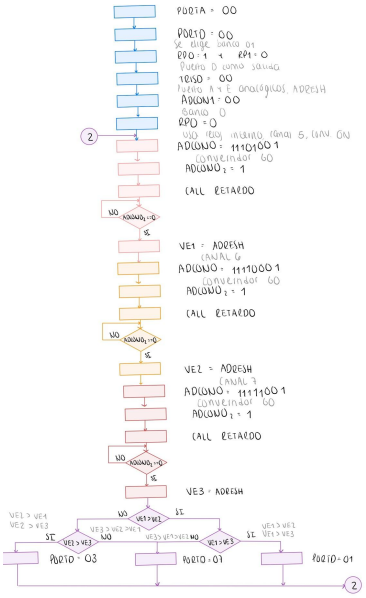
Voltaje	**Binario**
1 V	0000 1111
2 V	0001 1111
3 V	0010 1111
4 V	0011 1111
4.8 V	0110 1111

Con ellos, y con la configuración inicial mostrada anteriormente se realizó la siguiente carta ASM:



Ejercicio 3

Para el Ejercicio 3 se realizó la siguiente Carta ASM:



Primero se tiene la configuración inicial con la única diferencia que al modificar ADCON0 se elige el canal 5. Una vez en el canal correcto, se prende la bandera GO para iniciar la conversión, se llama un retardo y se verifica si la bandera es 0 o DONE. En caso de que lo sea, la conversión ya terminó y el valor de ADRESH se guarda en una variable VE1. En caso de que no haya acabado la conversión se espera hasta que acabe. El mismo proceso se realiza para el canal 6 y el canal 7 guardando sus valores en VE2 y VE3 respectivamente; esto se logra modificando el valor de ADCON0 antes de iniciar la conversión.

Una vez que se tienen los tres valores, se verifica primero si VE1 > VE2. En caso de serlo se verifica ahora que VE1 > VE3. Si esto es cierto, VE1 es el mayor y al Puerto D se le asigna un valor de 01. En caso contrario, VE3 es el mayor y se le asigna al Puerto D el valor de 07. Por otro lado si en la primera condición se tiene que VE1 < VE2, se verifica ahora si VE2 > VE3. Si esto es cierto, VE2 es el mayor y el Puerto D toma un valor de 03. En caso contrario, VE3 es el mayor y el Puerto D toma el valor de 07.

Después de la asignación de valor al Puerto D se vuelve a activar el convertidor en los tres canales para obtener tres nuevos voltajes a comparar.

Programas comentados

Ejercicio 1

```
processor 16f877
include<p16f877.inc>

VAL equ 0X20

    ORG 0               ;Vector de reset
    GOTO INICIO         ;Salto al inicio del programa

    ORG 5               ;Inicio del programa
INICIO:
    CLRF PORTA          ;limpia puerto A
    BSF STATUS,RP0      ;cambia a banco 1
    BCF STATUS,RP1
    CLRF TRISD          ;Configura puerto D como salida
    MOVLW B'00000000'
    MOVWF ADCON1        ;Configura puerto A y E como analogicos y el resultado se almacenara; en ADRESH
    BCF STATUS,RP0      ;cambia a banco 0
    MOVLW B'11111001'   ;W = 11111001
    MOVWF ADCON0        ;Configura ADCON0 con W: usara; reloj interno, canal 7, DONE, conv.A/D encendido
    CLRF PORTD          ;limpia puerto D
REPITE:
    BSF ADCON0,2        ;prende bit 2 de ADCON0 -> GO: inicia la comparacion
    CALL RETARDO        ;llama subrutina de retardo
ESPERA:
    BTFSC ADCON0,2      ;checa bit 2 de ADCON0
    GOTO ESPERA         ;Si es 1, regresa a la etiqueta ESPERA
    MOVF ADRESH,W       ;Si es 0, W = ADRESH
    MOVWF PORTD         ;PORTD = W
    GOTO REPITE         ;salta a la etiqueta REPITE
RETARDO
    MOVLW 0X30          ;W = 30
    MOVWF VAL           ;VAL = W
LOOP:
    DECFSZ VAL          ;decrementa en 1 a VAL, si es cero salta
    GOTO LOOP           ;VAL != 0, salta a la etiqueta LOOP
    RETURN              ;VAL = 0, regresa al flujo original, acaba subrutina

    END
```

Ejercicio 2

```
processor 16f877
include<p16f877.inc>

VAL equ 0X20
SAL0 equ B'00001111'    ;Valor registrado de 1V
SAL1 equ B'00011111'    ;Valor registrado de 2V
SAL2 equ B'00101111'    ;Valor registrado de 3V
SAL3 equ B'00111111'    ;Valor registrado de 4V
SAL4 equ B'01001111'    ;Valor registrado de 4.8V

    ORG 0               ;Vector de reset
    GOTO INICIO         ;Salto al inicio del programa

    ORG 5               ;Inicio del programa
INICIO:
    CLRF PORTA          ;limpia puerto A
    BSF STATUS,RP0      ;cambia a banco 1
    BCF STATUS,RP1
    CLRF TRISD          ;Configura puerto D como salida
    MOVLW B'00000000'
    MOVWF ADCON1        ;Configura puerto A y E como analogicos y el resultado se almacenara; en ADRESH
    BCF STATUS,RP0      ;cambia a banco 0
    MOVLW B'11111001'   ;W = 11111001
    MOVWF ADCON0        ;Configura ADCON0 con W: usara; reloj interno, canal 7, DONE, conv.A/D encendido
    CLRF PORTD          ;limpia puerto D
REPITE:
    BSF ADCON0,2        ;prende bit 2 de ADCON0 -> GO: inicia la comparacion
    CALL RETARDO        ;llama subrutina de retardo
ESPERA:
    BTFSC ADCON0,2      ;checa bit 2 de ADCON0
    GOTO ESPERA         ;Si es 1, regresa a la etiqueta ESPERA

    MOVLW SAL4
    SUBWF ADRESH,W
    BTFSS STATUS,C      ;ENTRADA < 4.8V?
    GOTO COMP4          ;if true: siguiente comparación
    MOVLW 0X05          ;else:
    MOVWF PORTD         ;SALIDA = 5
    GOTO REPITE         ;Comparar de nuevo

COMP4:
    MOVLW SAL3
    SUBWF ADRESH,W
    BTFSS STATUS,C      ;ENTRADA < 4V?
    GOTO COMP3          ;if true: siguiente comparación
    MOVLW 0X04          ;else:
    MOVWF PORTD         ;SALIDA = 4
    GOTO REPITE         ;Comparar de nuevo

COMP3:
    MOVLW SAL2
    SUBWF ADRESH,W
    BTFSS STATUS,C      ;ENTRADA < 3V?
    GOTO COMP2          ;if true: siguiente comparación
    MOVLW 0X03          ;else:
    MOVWF PORTD         ;SALIDA = 3
    GOTO REPITE         ;Comparar de nuevo


COMP2:
    MOVLW SAL1
    SUBWF ADRESH,W
    BTFSS STATUS,C      ;ENTRADA < 2V?
    GOTO COMP1          ;if true: siguiente comparación
    MOVLW 0X02          ;else:
    MOVWF PORTD         ;SALIDA = 2
    GOTO REPITE         ;Comparar de nuevo

COMP1:
    MOVLW SAL0
    SUBWF ADRESH,W
    BTFSS STATUS,C      ;ENTRADA < 1V?
    GOTO COMP0          ;if true: siguiente comparación
    MOVLW 0X01          ;else:
    MOVWF PORTD         ;SALIDA = 1
    GOTO REPITE         ;Comparar de nuevo

COMP0:
    CLRF PORTD          ;SALIDA = 0
    GOTO REPITE         ;salta a la etiqueta REPITE


RETARDO
    MOVLW 0X30          ;W = 30
    MOVWF VAL           ;VAL = W
LOOP:
    DECFSZ VAL          ;decrementa en 1 a VAL, si es cero salta
    GOTO LOOP           ;VAL != 0, salta a la etiqueta LOOP
    RETURN              ;VAL = 0, regresa al flujo original, acaba subrutina

    END
```

Ejercicio 3

```
processor 16f877
include<p16f877.inc>

VAL equ 0X20            ;Valor para el retardo
VE1 equ 0X21            ;Valor para primer potenciómetro
VE2 equ 0X22            ;Valor para segundo potenciómetro
VE3 equ 0X23            ;Valor para tercer potenciómetro

    ORG 0               ;Vector de reset
    GOTO INICIO         ;Salto al inicio del programa

    ORG 5               ;Inicio del programa
INICIO:
    CLRF PORTA          ;limpia puerto A
    CLRF PORTD          ;limpia puerto D
    BSF STATUS,RP0      ;cambia a banco 1
    BCF STATUS,RP1
    CLRF TRISD          ;Configura puerto D como salida
    MOVLW B'00000000'
    MOVWF ADCON1        ;Configura puerto A y E como analogicos y el resultado se almacenara; en ADRESH
    BCF STATUS,RP0      ;cambia a banco 0

REPITE:
    MOVLW B'11101001'   ;W = 11101001
    MOVWF ADCON0        ;Configura ADCON0 con W: usara; reloj interno, canal 5, DONE, conv.A/D encendido
    BSF ADCON0,2        ;prende bit 2 de ADCON0 -> GO: inicia la comparacion
    CALL RETARDO        ;llama subrutina de retardo
ESPERA1:
    BTFSC ADCON0,2      ;checa bit 2 de ADCON0
    GOTO ESPERA1        ;Si es 1, regresa a la etiqueta ESPERA
    MOVF ADRESH,W       ;Si es 0, W = ADRESH
    MOVWF VE1           ;VE1 = W

    MOVLW B'11110001'   ;W = 11110001
    MOVWF ADCON0        ;Configura ADCON0 con W: usara; reloj interno, canal 6, DONE, conv.A/D encendido
    BSF ADCON0,2        ;prende bit 2 de ADCON0 -> GO: inicia la comparacion
    CALL RETARDO        ;llama subrutina de retardo
ESPERA2:
    BTFSC ADCON0,2      ;checa bit 2 de ADCON0
    GOTO ESPERA2            ;Si es 1, regresa a la etiqueta ESPERA
    MOVF ADRESH,W       ;Si es 0, W = ADRESH
    MOVWF VE2           ;VE1 = W

    MOVLW B'11111001'   ;W = 11111001
    MOVWF ADCON0        ;Configura ADCON0 con W: usara; reloj interno, canal 7, DONE, conv.A/D encendido
    BSF ADCON0,2        ;prende bit 2 de ADCON0 -> GO: inicia la comparacion
    CALL RETARDO        ;llama subrutina de retardo
ESPERA3:
    BTFSC ADCON0,2      ;checa bit 2 de ADCON0
    GOTO ESPERA3        ;Si es 1, regresa a la etiqueta ESPERA
    MOVF ADRESH,W       ;Si es 0, W = ADRESH
    MOVWF VE3           ;VE1 = W

    MOVF VE2,W
    SUBWF VE1,W
    BTFSS STATUS,C      ;VE1 > VE2
    GOTO COMPNEG        ;if false: comparación donde VE1 es menor
    GOTO COMPPOS        ;if true: comparación donde VE1 es mayor

 COMPPOS:
     MOVF VE3,W
     SUBWF VE1,W
     BTFSS STATUS,C      ;VE1 > VE3
     GOTO V3             ;if false: VE3 es el mayor
     MOVLW 0X01          ;else:
     MOVWF PORTD         ;PORTD = 1
     GOTO REPITE         ;Realizar la comparación de nuevo
 COMPNEG:
     MOVF VE3,W
     SUBWF VE2,W
     BTFSS STATUS,C      ;VE2 > VE3
     GOTO V3             ;if false: VE3 es el mayor
     MOVLW 0X03          ;else:
     MOVWF PORTD         ;PORTD = 3
     GOTO REPITE         ;Realizar la comparación de nuevo
 V3: ;Se realiza cuando VE3 es el mayor
     MOVLW 0X07
     MOVWF PORTD         ;PORTD = 7
     GOTO REPITE         ;Realizar la comparación de nuevo

 RETARDO
     MOVLW 0X30          ;W = 30
     MOVWF VAL           ;VAL = W
 LOOP:
     DECFSZ VAL          ;decrementa en 1 a VAL, si es cero salta
     GOTO LOOP           ;VAL != 0, salta a la etiqueta LOOP
     RETURN              ;VAL = 0, regresa al flujo original, acaba subrutina

     END
```

Conclusiones y/o comentarios

Zepeda Baeza Jessica:

Esta práctica se enfocó en el uso de entradas analógicas que guardan el valor del voltaje suministrado. El primer ejercicio trató de las instrucciones básicas que debe llevar un programa que convierte una entrada a un valor analógico, configurando adecuadamente los registros ADCON0 y ADCON1, prendiendo y revisando la bandera GO/DONE y obteniendo el valor de ADRESH o ADRESL. Con este programa fue posible realizar ejercicios más complejos como tener un valor de salida para cierto rango de valores analógicos o activar el convertidor en distintos canales y guardar su valor en otros registros para después utilizarlos. Esta práctica permitió que no solo podamos y sepamos utilizar datos digitales en un circuito sino también analógicos, lo que por ejemplo puede tener aplicaciones para indicar la carga en una pila. Además la construcción de los programas fue más sencilla ya que estamos más familiarizados con el conjunto de instrucciones, los saltos que verifican un bit y el valor de la bandera Carry que indica si un número es mayor o menor al hacer una resta de dos números.

Barreiro Valdez Alejandro:

En esta práctica se utilizaron las entradas analógicas que son capaces de mostrar el valor de voltaje que se tiene de entrada. Para el primer ejercicio se requirió hacer un convertidor simple que permitió convertir la entrada a un valor digital. Se configuraron los registros de ADCON0 y ADCON1, así como la bandera de GO/DONE para configurar el convertidor. Una vez realizado un convertidor analógico/digital básico se realizaron ejercicios más complejos. Para el segundo ejercicio se realizó un código donde se mostraba un valor específico si el voltaje suministrado se encontraba en cierto rango. Para ello se obtuvieron los valores digitales que equivalen a cada uno de los valores analógicos. El siguiente ejercicio consistió en utilizar tres entradas para convertir cada una de las entradas y compararlas. En este caso se convierte el valor analógico a digital, se opera y posteriormente se ofrece un resultado. Esta operación es de gran utilidad para analizar datos del mundo real y solo se puede realizar gracias al convertidor analógico/digital.