ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica y Electrónica

Carrera de Electrónica y Automatización

SISTEMAS BASADOS EN MCU

Práctica 3.4

MANEJO DE DISPLAYS EN LENGUAJE C PARA EL PIC16F877

Autor:

Iza Tipanluisa Alex Paul

Docente:

Ing. Amparo Meythaler

NRC: 4891

1) OBJETIVOS

- Identificar la forma en la que están elaborados los teclados matriciales.
- Identificar la forma de implementar y programar un teclado matricial con el PIC16F877.

2) MARCO TEORICO

MANEJO DE DISPLAYS

El display led de 7 segmentos es un componente electrónico que me permite visualizar un valor numérico para una determinada aplicación. Cuando se quiere mostrar datos en el display, existen dos opciones para hacerlo, una utilizar un decodificador BCD a 7 segmentos después del microcontrolador, y otra es generar con el mismo microcontrolador el código 7 segmentos equivalente a cada número de 0 a 9.

Para que el microcontrolador maneje el display de 7 segmentos es necesario hacer la siguiente conexión (Para un display 7 segmentos de cátodo común). La misma conexión puede hacerse para un display led 7 segmentos de ánodo común, la diferencia es que el común va a positivo y en la tabla de abajo intercambiar 1 por 0 y 0 por 1.

En lenguaje C es común utilizar para estas aplicaciones un arreglo. Para esto en el programa se crea una constante tipo array que almacenan los valores codificados. Ejemplo: const int x[]= {64,121,36,48,25,18,2,120,0,16}. Los datos pueden sacarse al display utilizando un lazo for..

3) EQUIPOS Y MATERIALES

- PC con el paquete PIC C
- PC con el paquete PROTEUS.

4) ACTIVIDADES

1) Trabajo Preparatorio:

a) Realice el diagrama de flujo, la codificación y la implementación correspondientes de un programa que presente en dos displays mediante la técnica Decodificación Simple un contador (el mismo ejercicio de leds de la práctica anterior). No usar un circuito integrado extra como decodificador.

b) Verifique el funcionamiento del MPLAB IDE y del PROTEUS.

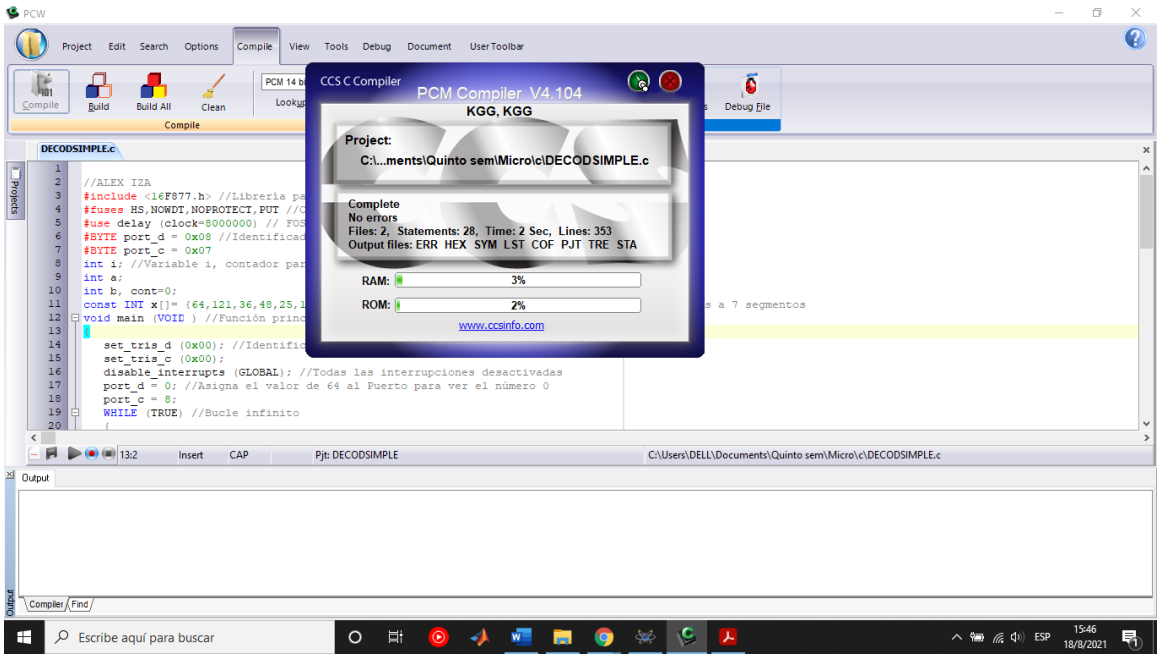

Ilustración 1, Compilación del programa

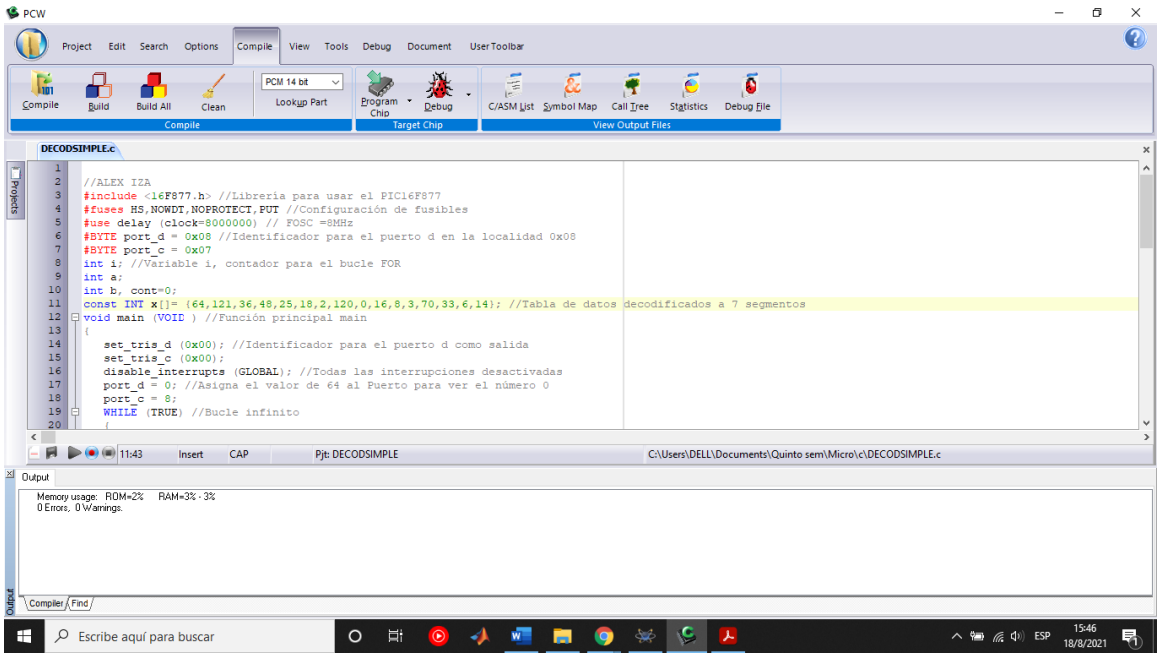

Ilustración 2, Compilación exitosa, 0 errores

5) RESULTADOS

- **Explique los errores cometidos en el ejercicio realizado (si los tuvo) y la forma de corregirlos.**

  Tuve un error al mostrar el dato ya que en las decenas me contaba dos veces el dato A cuando ya debía cambiar al sato B; lo solucioné cambiando el rango del For de i=11 a i=10.

- **Explique dónde se establece dentro del software que el display utilizado es de ánodo o de cátodo común.**
  Se establece al inicio en el arreglo, en nuestro caso

  const INT x[]= {64,121,36,48,25,18,2,120,0,16,8,3,70,33,6,14};

  los valores son los que definen si los displays son cátodos o ánodos en este caso ánodo común

6) DISEÑO

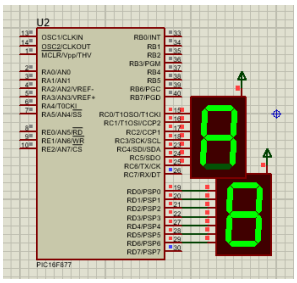

Ilustración 3, Visualización del valor A8

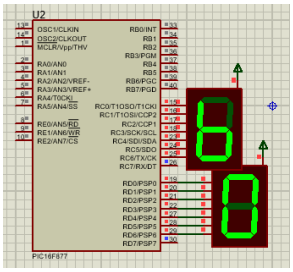

Ilustración 4, Visualización del valor B0

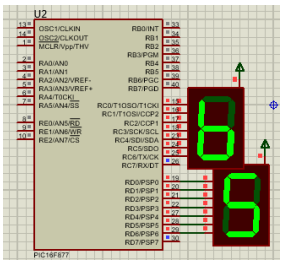

Ilustración 5, Visualización del valor B5

7) DIAGRAMA DE FLUJO

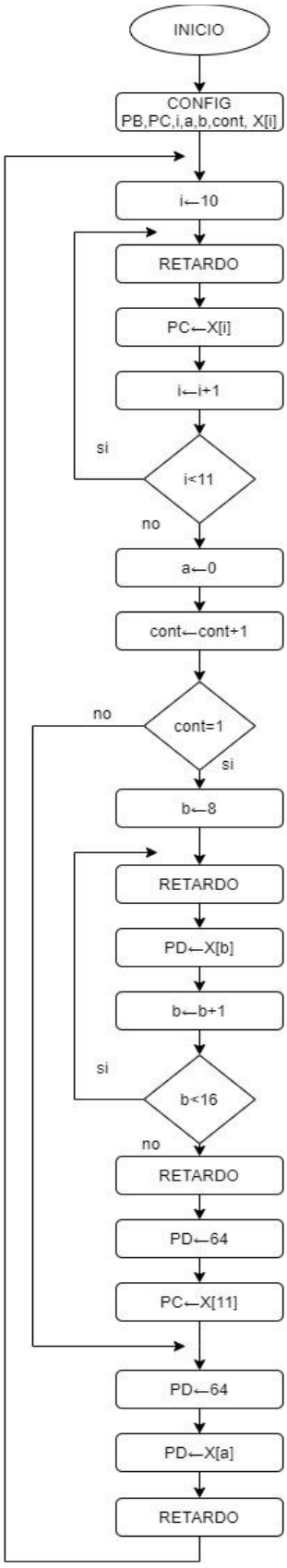

Diagrama 1, Programa

8) PROGRAMA

```
//ALEX IZA

#include <16F877.h> //Librería para usar el PIC16F877

#fuses HS,NOWDT,NOPROTECT,PUT //Configuración de fusibles

#use delay (clock=8000000) // FOSC =8MHz

#BYTE port_d = 0x08 //Identificador para el puerto d en la localidad 0x08

#BYTE port_c = 0x07

int i; //Variable i, contador para el bucle FOR

int a;

int b, cont=0;

const INT x[]= {64,121,36,48,25,18,2,120,0,16,8,3,70,33,6,14}; //Tabla de datos
decodificados a 7 segmentos

void main (VOID ) //Función principal main

{

  set_tris_d (0x00); //Identificador para el puerto d como salida

  set_tris_c (0x00);

  disable_interrupts (GLOBAL); //Todas las interrupciones desactivadas

  port_d = 0; //Asigna el valor de 64 al Puerto para ver el número 0

  port_c = 8;

  WHILE (TRUE) //Bucle infinito

  {

    cont=0;

    FOR (i=10; i<=10;++i)

    {

      delay_ms(500);

      port_c= x[i];

       FOR (a=0; a<=5;++a)
```

```
{
cont=cont+1;
     if(cont==1){
     FOR ( b=8; b<=15;++b)
      {
         delay_ms(500); //Retardo de 500 ms.
         port_d= x[b];
           }
           delay_ms(500);
         port_d= 64;
         port_c= x[11];
        }
      delay_ms(500);
   port_d= x[a];
   }
delay_ms(500);
} }  }
```

9) CONCLUSIONES

- Se ha concluido que si no se sabe el tipo del display; al momento de conectarle en un ánodo común si se necesita el número cero aparecerá una línea si es cátodo común, por lo cual se deberá cambiar.

- Se ha concluido que la programación en C es muy diferente de la programación en ensamblador; pero el diagrama de flujo es general, por lo cual se puede usar con cualquier lenguaje de programación.

- Se ha concluido que en el lenguaje de programación C no es necesario cambiarnos de banco para usar los TRIS

10) RECOMENDACIONES

- Se recomienda revisar los usos correctos de los comandos en C ya que si lo usa incorrectamente; no se obtendrán los resultados requeridos

- Se recomienda verificar el funcionamiento del programa con todos los datos que podrían generar problemas.

- Se recomienda usar variables con el nombre de la función que va a realizar para que no se generen complicaciones y confusiones en su uso

11) BIBLIOGRAFIA

Meythaler, A. (2021). Sistemas basados en MCU. Ecuador: UFA ESPE