NOMBRE DEL ALUMNO: _____
Apellido Paterno

_____
Apellido Paterno Nombre

N° DE BOLETA: _____ GRUPO: _____

ASIGNATURA: *Micro Electrónica Programable*

HOJA	DE

FECHA		
DIA	MES	AÑO

EVALUACION

PROFESOR: _____

Práctica 3
Gestión de puertos de entrada

Competencias de La Unidad:

- Emplea el microcontrolador en la comunicación de datos y control de periféricos de forma multiplexada.

Resultado de Aprendizaje Propuesto (RAP):

- Usa diferentes elementos periféricos utilizando los puertos del microcontrolador de forma multiplexada.
- Utiliza los puertos del microcontrolador de forma bidireccional en la solución de un problema.

.

Objetivos de la Práctica:

1. Realizar la simulación de un programa para comprobar su funcionamiento utilizando herramientas computacionales
2. Desarrolla programas que obtengan valores de entrada del microcontrolador.
3. Identificar las funciones de entrada de datos
4. Manejo de arreglos y sentencias de control en lenguaje C
5. Realizar un control de elementos visuales tipo LED (Display 7 segmentos), mediante elementos de entrada.
6. Implementar un programa en un circuito basado en microcontrolador.

Equipo Necesario	*Material Necesario*
Computadora (con el Software MPLAB IDE, IC-PROG o similar, compilador C, Simulador de circuitos electrónicos “Proteus”) Programador tipo JDM o similar.	Instrucciones del PIC 16F887 u otro de gama media o alta MIcrocontrolador PIC16F887 u otro de gama media o alta Capacitores Display de cátodo común Resistencias pushbutton Cristal de cuarzo de 4MZ (Para los valores de estos elementos ver figura 3.1)

Introducción Teórica

Puertos de Entrada y salida

El Microcontrolador 16F887/884 cuenta con 5 puertos direccionales, denominados A, B C,D y E y la versión 16F882/883/886 cuenta solo con los puertos denominados A,B y C.

Sus principales características son:

- Programables como entradas o salidas individualmente.
- Capaces de trabajar con corrientes de 25 mA. en cada línea. No obstante la corriente total en los puertos A, B y E no puede superar los 200 mA. y en los puertos C,D otros 200 mA.
- Entradas tipo TTL o ST (Schmitt Trigger).
- Resistencias Pull-up (habilitadas por programa) en el puerto B

Las líneas de E/S en el PIC 16F887 están agrupadas en 5 puertos: A (8 bits), B (8 bits), C (8 bits). D (8 bits) y E (4 bits) Cada puerto de E/S tiene asociados dos registros TRISX y PORTX. El primer registro dispone de un bit por cada línea del puerto, y controlará si funciona como entrada (Input 1) o como salida (Output, 0). El segundo registro nos permite acceder al puerto para realizar una lectura de los bits, así como modificar los bits configurados como salida. Si se realiza una escritura y de forma inmediata una lectura, puede que la salida no haya alcanzado el nivel adecuado generando incoherencias

.

Para inicializar los puertos se realiza las siguientes instrucciones en ensamblador

Como entrada

- Colocarse en el banco 1,
- Cargar “1’s” en el registro TRISX, si se desea que se comporte el puerto como entrada, si solo desea configurar como entrada bits individuales colocar 1 en los bits que desee.
- Regresar al banco 0.

Como Salida

- Colocarse en el banco 1.
- Cargar “0’s” en el registro TRISX, si se desea que se comporte el puerto como salida, si solo desea configurar como salida bits individuales colocar 0 en los bits que desee.
- Regresar al banco 0

Nota: Para cambiar de banco se utiliza los bits 5 y 6 del registro de estado. Al cambiar al banco 1, configure todos los puertos. En el caso del PIC16F887, debe configurarse los pines del Puerto A como entradas digitales colocan el valor 00H en el registro ANSEL Y ANSELH

En el caso del compilador de C, los pasos anteriores se efectúan utilizando las directivas #use standard_io(X), #use fast_io(X) y #use fixed_io(X) y la instrucción set_tris_X(valor).

Instrucciones de lectura y salida de datos utilizando directivas del compilador:
Con el fin de obtener y enviar datos al exterior, el complilador C cuenta con diversas directivas e instrucciones para lograr este objetivo, enseguida mencionamos las más relevantes.

Directivas de entrada-salida

#USE STANDARD_IO (puerto)
Esta directiva afecta al código que el compilador genera para las instrucciones de entrada y salida.. El método causará que el compilador genere el código para hacer que un pin de I/O sea entrada o salida cada vez que se utiliza.

Ejemplo:
#use standard_io(b)

#USE FAST_IO (puerto)
Las funciones “input” y “output” no reprograman los pines de los puertos cada vez que se utilizan, provocando que se genere un menor código, es importante considerar que previamente se debe configurar los puertos con las instrucción set_tris_X(valor).

#use fast_io(A)

- ***#USE FIXED_IO (puerto_OUTPUTS=pin_x#, pin_x#...)***

Las funciones “input” y “output” reprograman los pines de los puertos cada vez que se utilizan, pero con la característica que los puertos se definen de acuerdo a la información que indica la directiva (donde solo se indican las terminales de salida) sin tener en cuenta si la operación es de entrada o de salida.

Ejemplo:
#use fixed_io(a_outputs=PIN_A2 ,PIN_A3)”

Instrucciones de entrada

Existen instrucciones para leer la información del exterior por los puertos se describen como siguen:

INPUT(pin)
Devuelve el estado '0' o '1' de la terminal indicada en pin. El método de acceso de I/O depende de la última directiva #USE *_IO utilizada. El valor de retorno es un entero corto

INPUT_X()
Obtiene el valor de puerto correspondiente y puede ser asignado a una variable. El registro de configuración se cambia de acuerdo a lo especificado en la directiva #USE *_IO, que por defecto es standard_IO, pero en caso de no usarla es necesario configurar el registro adecuado para que efectué la función de entrada

Instrucciones de salida

Las instrucciones para enviar información hacia exterior son las siguientes:

OUTPUT_X(Valor)	Envía el valor correspondiente (0-255) al puerto indicado en X
OUTPUT_BIT (PIN,VALOR)	Envía el valor 0 o 1 al pin especificado(0-7) del puerto
OUTPUT_LOW(PIN)	Envía el valor 0 al pin especificado(0-7) del puerto
OUTPUT_HIGH(PIN)	Envía el valor 1 al pin especificado(0-7) del puerto

ACTIVIDADES TEÓRICAS PREVIAS
Investigar los siguientes conceptos:

- Investigue las diferentes formas de gestionar o utilizar los puertos en lenguaje C y cuales son sus ventajas o desventajas.
- Display de 7segmentos
- Instrucciones para modificar el contenido de los registros de configuración de puertos.
- Instrucciones de entrada y salida de datos en el compilador de CCS
- Instrucciones de retardo en el compilador CCS
- Tipos de datos y operadores en lenguaje C
- Sentencias *if , while, for* de lenguaje C.

ACTIVIDADES PREVIAS

- *Crear un proyecto de nombre pra3 en la carpeta c:\MEPIC\practica3 en MPLAB o PIC C Compiler. Los programas de cada ejercicio deben ser guardados con el nombre practica3X.c con X= 1, 2, 3…,A.*

- *En el caso de utilizar MPLAB, realizar los siguientes pasos:*

a. *Utilizar Project wizard y seleccionar el compilador de c*
b. *Agregar al proyecto los archivos adecuados con extensión c y h.*
c. *Habilitar Simulador MPLAB SIM y modificar la frecuencia del simulador a 4 Mhz.*
d. *Utilizaremos la herramienta de stopwatch, para obtener la elija Debugger >> Stopwatch.*
e. *Obtener la herramienta de watch, de la siguiente manera View>> watch.*
f. *Y seleccione los registros PORTA, PORTB, PORTC, PORTD, PORTE, TRISA, TRISB, TRISC, TRISD, TRISE y W*

- *Si usa PIC C compiler crear el proyecto únicamente e incluir un archivo nuevo.*

ACTIVIDADES PRÁCTICAS

Parte 1

1. *Armar el siguiente circuito*

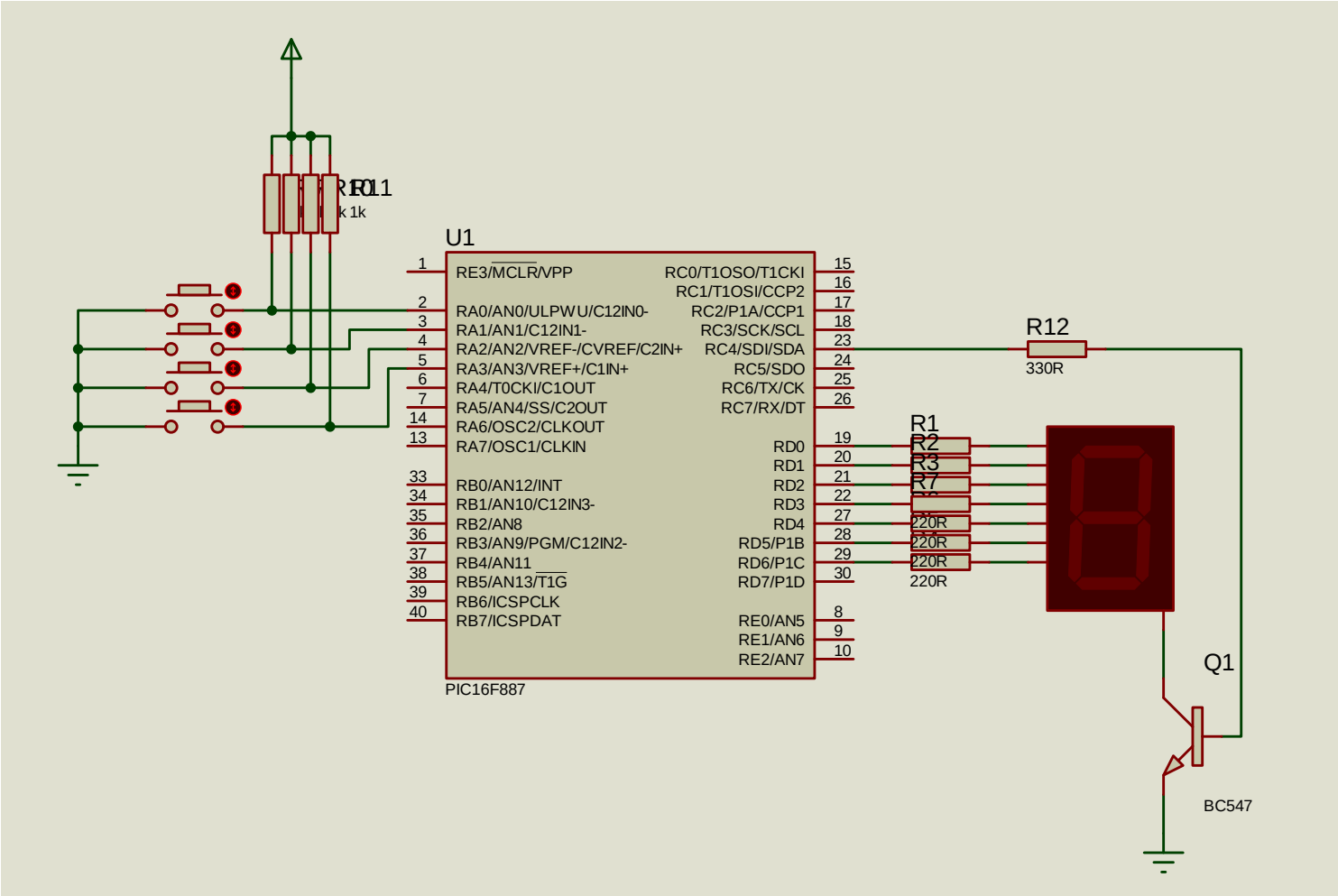

Figura 3.1

Nota: *La terminal 12 o 31 del PIC16F887 se conectan a tierra.*

Realizar el siguiente programa y observar su funcionamiento utilizando la opción watch y stopwatch del simulador MPLAB (en caso de usarlo) y posteriormente grabarlo en el circuito de la figura 3.1, así como simularlo en PROTEUS

Ejemplo 1

```
#include <16f887.h>
#use delay(clock=4000000)
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use standard_io(d)
#use standard_io(c)
#use standard_io(a)
void main()
{
output_c(0x10);
while(true)
{
int a,x=0;
output_d(0xFF);
if(input(PIN_A0)==0)
{
for(a=0;a<=3;a++)
{
output_d(0x7F);
delay_ms(500);
output_d(0x00);
delay_ms(500);
}
}
else if(input(PIN_A1)==0)
{
for(a=0;a<10;a++)
{ x=x+(2*a+1);
output_d(x);
delay_ms(500);
output_d(0);
}
}
}
}
```

- *Nota: A grabarlo deshabilitar WDT y LVP y habilitar PWRTE y BODEN en la palabra de configuración, además recuerde seleccionar el tipo de oscilador a XT*

Ejemplo 2

```
#include <16f887.h>
#use delay(clock=4000000)
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use standard_io(d)
#use standard_io(c)
#use standard_io(a)
byte CONST unidad[10] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
```

```
int i,j;
void main()
{
output_c(0x10);
i=0;
j=0;
while(true)
{
i=0;
j=0;
output_d(0x3F);
delay_ms(500);
output_d(0x00);
delay_ms(500);
if(input(PIN_A0)==0)
{
delay_ms(100);
while(i<=9)
{
output_d(unidad[i]);
delay_ms(500);
i++;
}
}
else if(input(PIN_A1)==0)
{i=9;
delay_ms(100);
while(j<=9)
{
output_d(unidad[i]);
delay_ms(500);
j++;
i--;
}
}
}
}
```

Ejemplo 3

```
#include <16f887.h>
#use delay(clock=4000000)
#fuses NOWDT,INTRC,NOLVP
#use standard_io(d)
#use standard_io(c)
#use standard_io(a)
int i;
void main()
{
output_c(0x10);
while(true)
{
```

```
output_d(0x0f);
i=input_A()&0X0F;
delay_ms(100);

switch(i)
{
case 14:
{output_d(0x06);
delay_ms(1000);
output_d(0x00);
break;}
case 13:
{output_d(0x5B);
delay_ms(1000);
output_d(0x00);
break;}
case 11:
{output_d(0x4F);
delay_ms(1000);
output_d(0x00);
break;}
case 7:
{output_d(0x66);
delay_ms(1000);
output_d(0x00);
break;}
default:
output_d(0x0f);
break;
}
}
```
}*Parte 2 Ejercicios*

1) **Realizar programa que efectué un conteo de forma ascendente, incrementando de uno en uno al presionar un botón conectado a RA0 y mostrándolo en un display de 7 segmentos conectado al puerto D**

2) ***Crear un programa que realice un dado electrónico que:***

Cuente del 0 al 6 y al presionar el interruptor conectado a RA0 se mantenga el valor durante 2 segundos y continúe la cuenta.

1. *Conclusiones*

   A. *Realizar conclusiones de manera individual.*

2. Cuestionario

a) ¿Qué función tiene la sentencia if ?
b) ¿Qué función tiene la sentencia for ?
c) ¿Qué función tiene la sentencia while ?
d) ¿Qué función tiene la sentencia switch ?
e) ¿Cuál es la instrucción para leer datos por puerto?
f) Menciona que función tiene la directiva y `#use standard_io(b)`
g) Como se define un arreglo en lenguaje C
h) Menciona los tipos de datos en lenguaje C

Comentarios Finales

- El alumno entrega un reporte de la práctica, como el profesor lo indique.
- El reporte debe contener el diagrama de flujo o algoritmo (Seudo código) de cada uno de los programas.
- Además, en el reporte deben anexarse las conclusiones y cuestionario contestado.