CLASE1: INTRODUCCION A LA PROGRAMACION EN MICROCONTROLADORES PIC Y MANEJO DE PUERTOS DE ENTRADA Y SALIDA

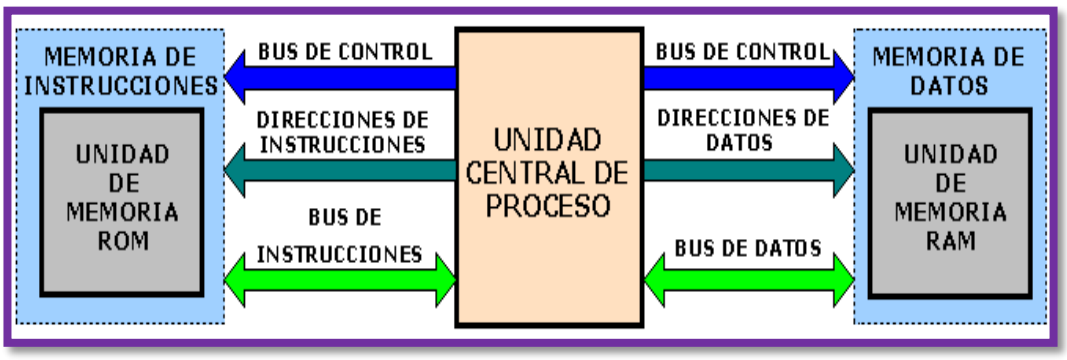
1. INTRODUCCION A LOS MICROCONTROLADORES
   1.1. DEFINICION:

   El microcontrolador PIC es un circuito integrado programable que pertenece a la familia de microcontroladores tipo RISC (Reducido conjunto de instrucciones de computación) lo cual facilita su programación.
   Las aplicaciones e información que poseen los microcontroladores PIC en internet son variadas lo cual lo hace el microcontrolador por excelencia para aplicaciones puntuales de bajo costo.

   La empresa que fabrica los microcontroladores PIC es Microchip Technology.

   1.2. ARQUITECTURA DE LOS MICROCONTROLADORES

   La arquitectura de los microcontroladores PIC es la arquitectura Harvard el cual posee 2 memorias separadas, la memoria de datos y la memoria de programa.
   Al poseer memorias separas lo convierte una microcontrolador rápido a diferencia de la arquitectura von Newman, que su memoria de datos y memoria de programa están juntas.



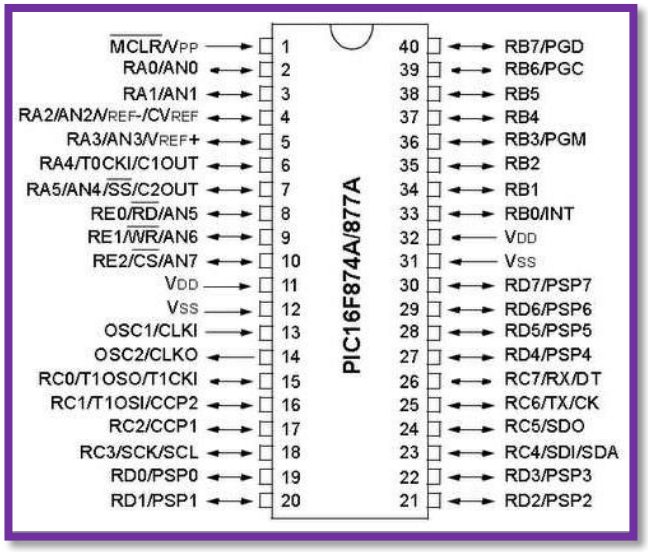
   1.3. GAMA DE LOS MICROCONTROLADORES

   Los microcontroladores PIC se clasifican en tres grandes grupos:

   - Gama base: 33 instrucciones de 12 bits de longitud.
     - ✓ MCU: 10FXXX/12F5XX/16F5X.
   - Gama media: 35 instrucciones de 14 bits de longitud.
     - ✓ MCU: 12FXXX/16FXXX.
   - Gama mejorada: 77 instrucciones de 16 bits de longitud.
     - ✓ MCU: 17CXXX/17CRXX/18FXXXX.

1.4. CARACTERISTICAS DE LOS MICROCONTROLADORES

- Velocidad de operación hasta 20Mhz.
- Memoria de programa tipo flash 8Kx14.
- Memoria de datos de 368 bytes.
- Memoria EEPROM de 256 bytes.
- 33 puertos de Entrada/Salida (Puerto A, B, C, D, E).
- 3 timers (timer0 a 8 bits, timer1 a 16 bits y timer2 a 8 bits).
- 8 canales de A/D.
- 2 puertos CCP (comparador, captura y PWM).
- Resistencias PULL-UP programable en el puerto B.
- 14 fuentes de interrupciones.
- Módulo de comunicación USART/SCI.



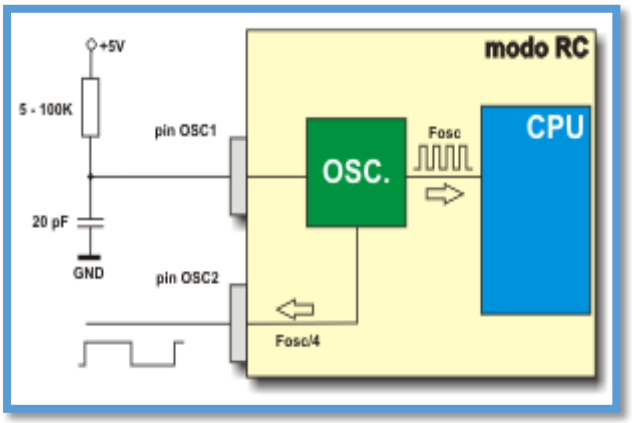
1.5. CONFIGURACION DEL CLOCK

1.5.1. CONFIGURACION DE LOS FUSES

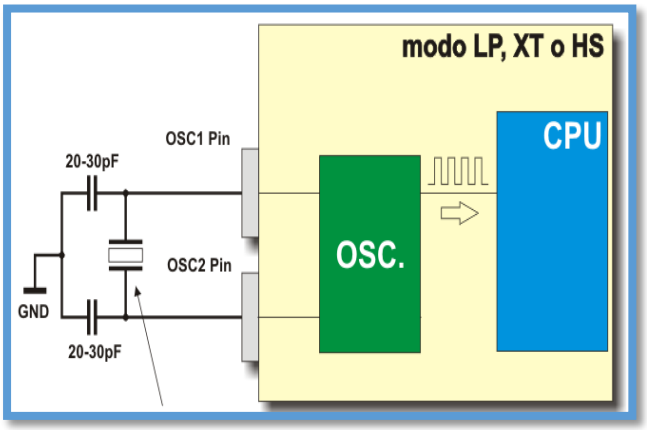
Oscilador	Descripción
LP	Cristal de bajo consumo de potencia
XT	Cristal/Resonador
HS	Cristal/Resonador de alta velocidad
RC	Resistencia-condensador

1.5.2. CONFIGURACION EN EL HARDWARE

1.5.2.1. CONFIGURACION RC



1.5.2.2. CONFIGURACION CRISTAL DE CUARZO

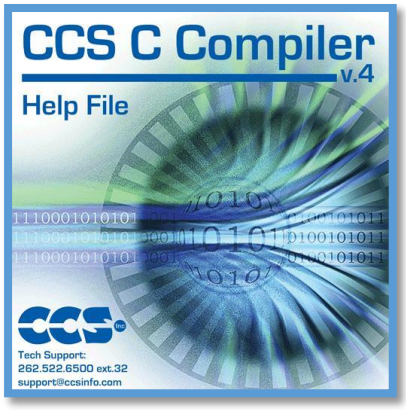


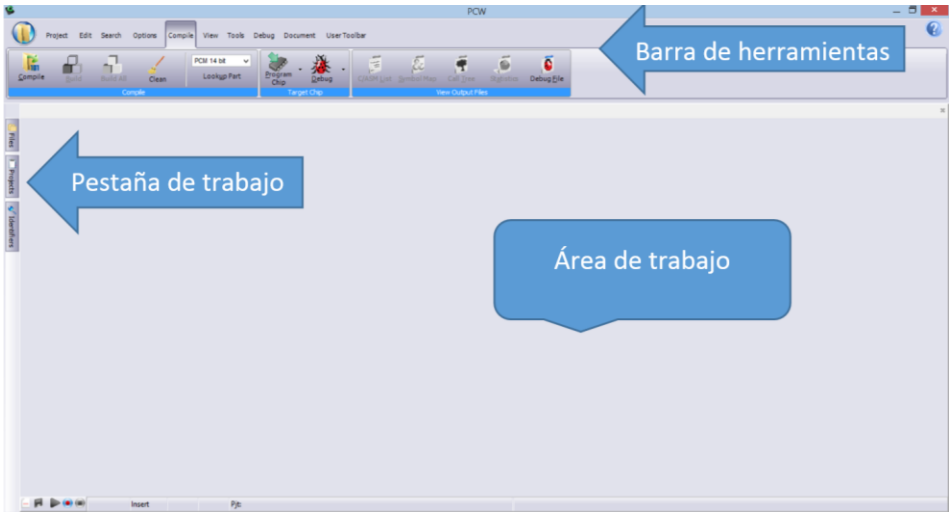
1.5.2.3.

1.6.

2. DESCRIPCION DEL SOFTWARE

El programa a utilizar será CCS C COMPILER, este software es el más usado para programar microcontroladores en leguaje C debido a que contiene una gran variedad de librería que facilita la programación.





2.1. PRIMEROS PASOS EN CCSCOMPILER

Para programar en C compiler primero se debe conocer cómo se estructura el programa.

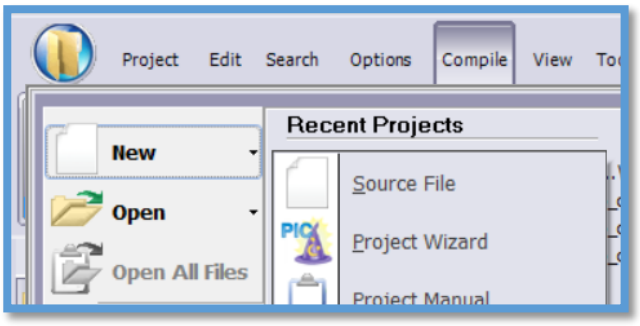
Estructura del programa:

- Directivas.
- Declaración de variables y constantes globales.
- Funciones e interrupciones.
- Función principal.
- Sentencias

Ahora que ya conocemos el entorno de programación y sabemos cómo se estructura un programa crearemos nuestro primer programa.

2.1.1. PASO1

Crear un nuevo Source File.



Se abrirá una ventana donde guardaremos nuestro programa, normalmente se crea una carpeta en el escritorio y se almacena en dicha carpeta.

2.1.2. PASO2

En la hoja en blanco que creamos, escribiremos el siguiente código:

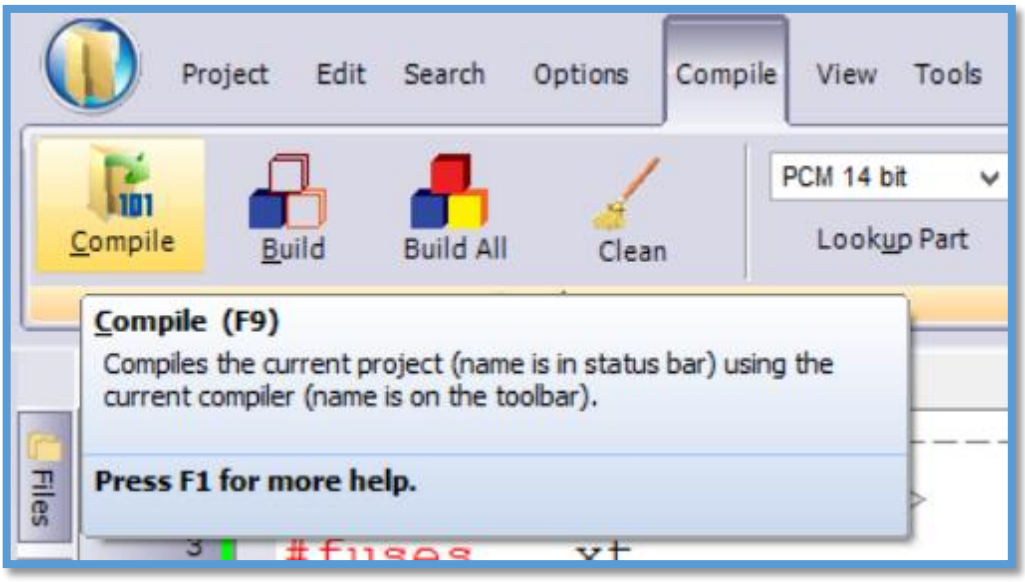
```
nkLed.c *
1  //-------DIRECTIVAS--------
2  #include <16f877a.h>        //Directiva que define el PIC a utilizar
3  #fuses    xt                //Directiva que define el crital como oscilador
4  #use      delay(clock=4M)   //Directiva que define la frec. de oscilacion
5
6  //----FUNCION PRINCIPAL----
7  void main()
8  {
9     set_tris_b(0b00000000); //Directiva el puerto B como salida
10    output_b(0b00000000);   //Los pines de puerto B en nivel bajo
11    while(true)
12    {
13       output_b(0b00000001);//Pin B0 en nivel alto
14       delay_ms(1000);      //Retardo de 1 segundo
15       output_b(0b00000000);//Pin B0 en nivel bajo
16       delay_ms(1000);      //Retardo de 1 segundo
17    }
18 }
```

*Las letras en gris son comentario y se definen con doble diagonal "**//**", los comentario no interviene directamente con el programa solo sirve como guía para el programador.*

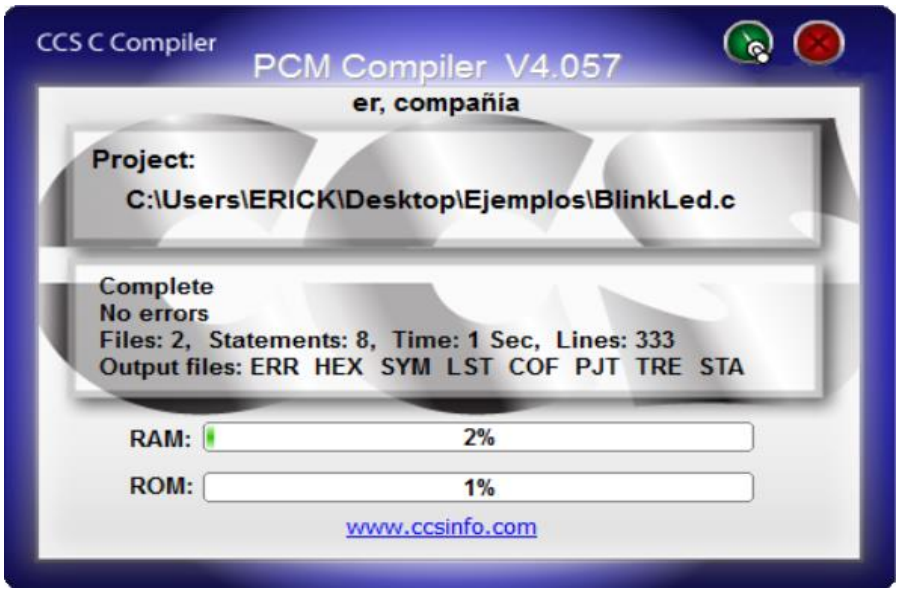
2.1.3. PASO3

Para finalizar la programación deberemos compilar el programa para verificar que no existan errores, además al compilar se genera un archivo con extensión ".HEX" que nos sirve para pasar nuestro código a de la PC al microcontrolador.

Presionamos la opción Compile



Aparecerá una ventana azul y genera el archivo “.HEX”



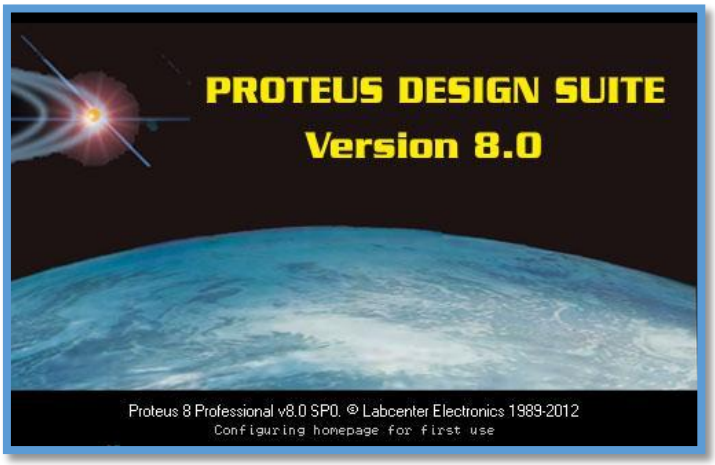
2.1.4.

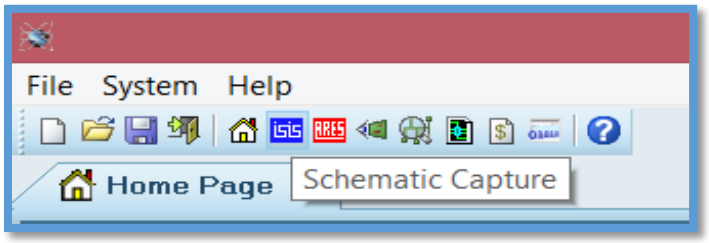
2.2. PASO2

2.3. PASO3

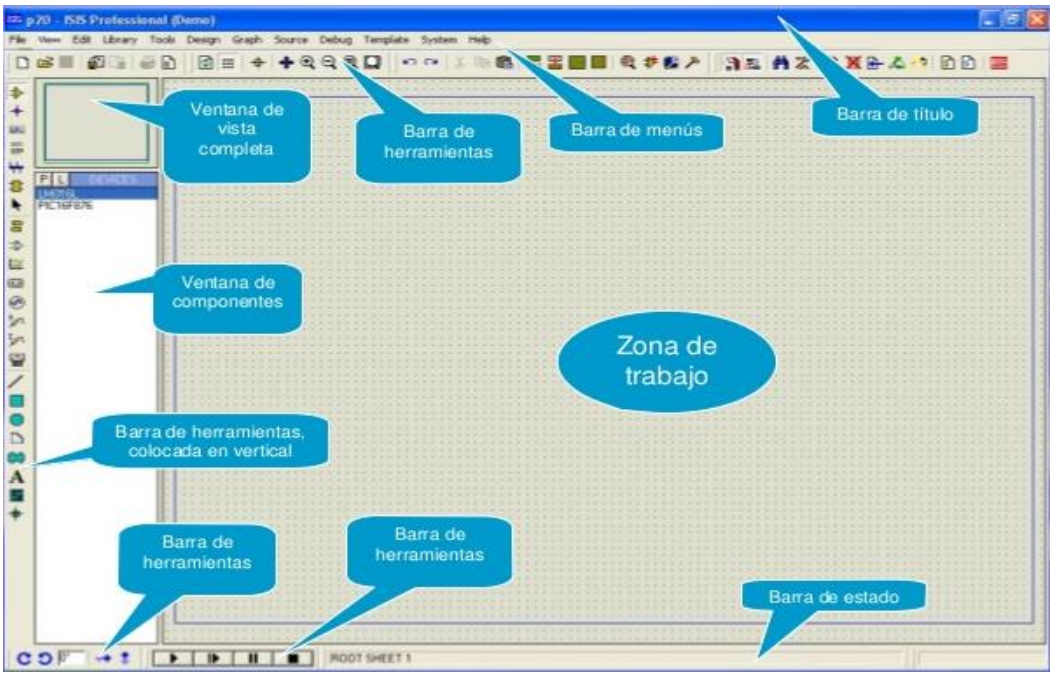
3. DESCRIPCION DEL SIMULADOR

   El programa a utilizar será PROTEUS, este software de simulación es el más usado debido a que contiene una gran variedad componentes como resistencias, condensadores, diodos, teclados matriciales, LCD, variedad de microcontroladores, motores, etc.





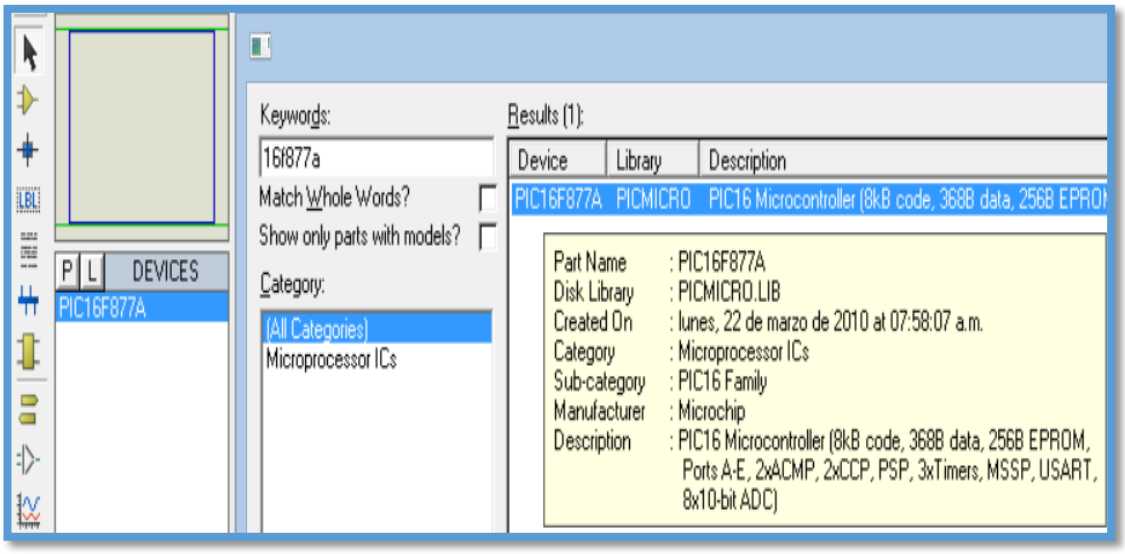
Al seleccionar el símbolo azul ISIS se abrirá la ventana de trabajo donde se describe en la siguiente imagen:



Para trabajar en Proteus debemos seguir los siguientes pasos:

3.1. PASO1

En la ventada de componentes seleccionamos el botón con la letra P, se abrirá una ventana donde escribiremos en Keywords 16f877a.

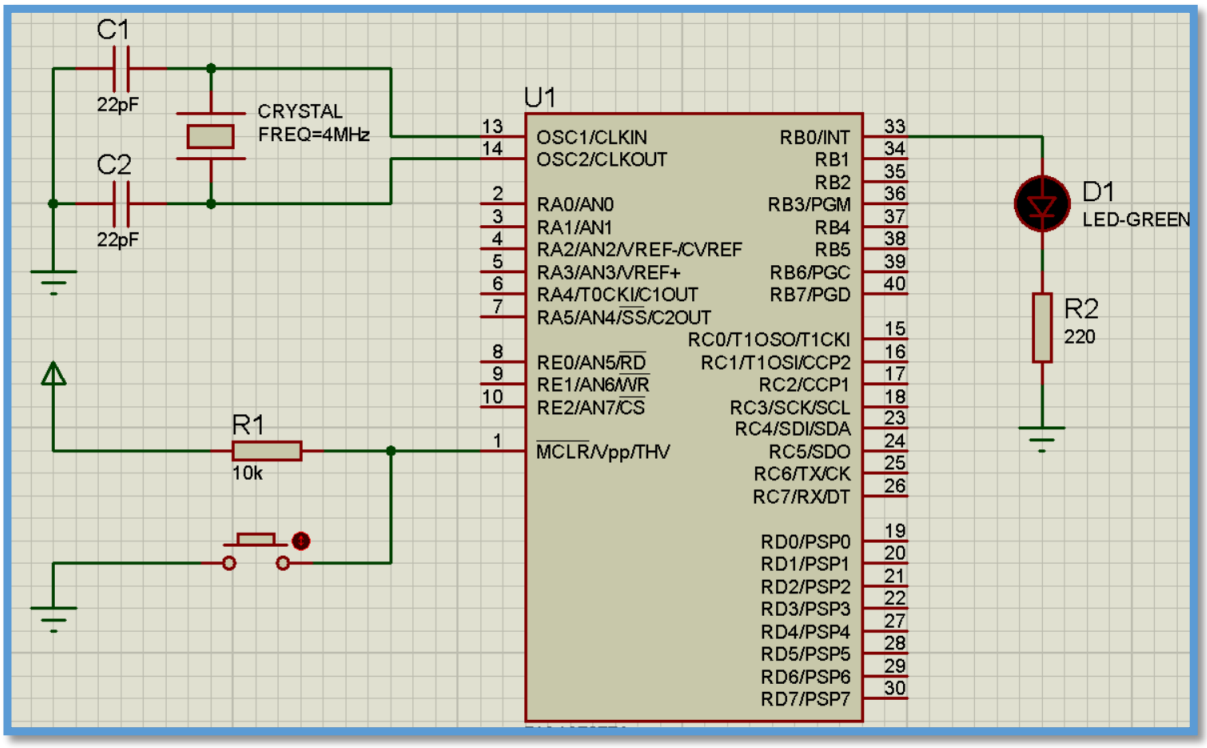


El componente aparecerá en Results, lo seleccionaremos con doble click y ahora lo tendremos disponible en la ventana de componentes.
Los componentes más usados son:

Componentes	Keyword Proteus
Resistencia	res
Condensador cerámico	Cap
Potenciómetro	Pot-hg
Batería	Battery
PIC 16F877A	16f877a
Teclado matricial	Keypad
LCD 16x2	LM016L
Cristal de cuarzo	crystal
Led	Led

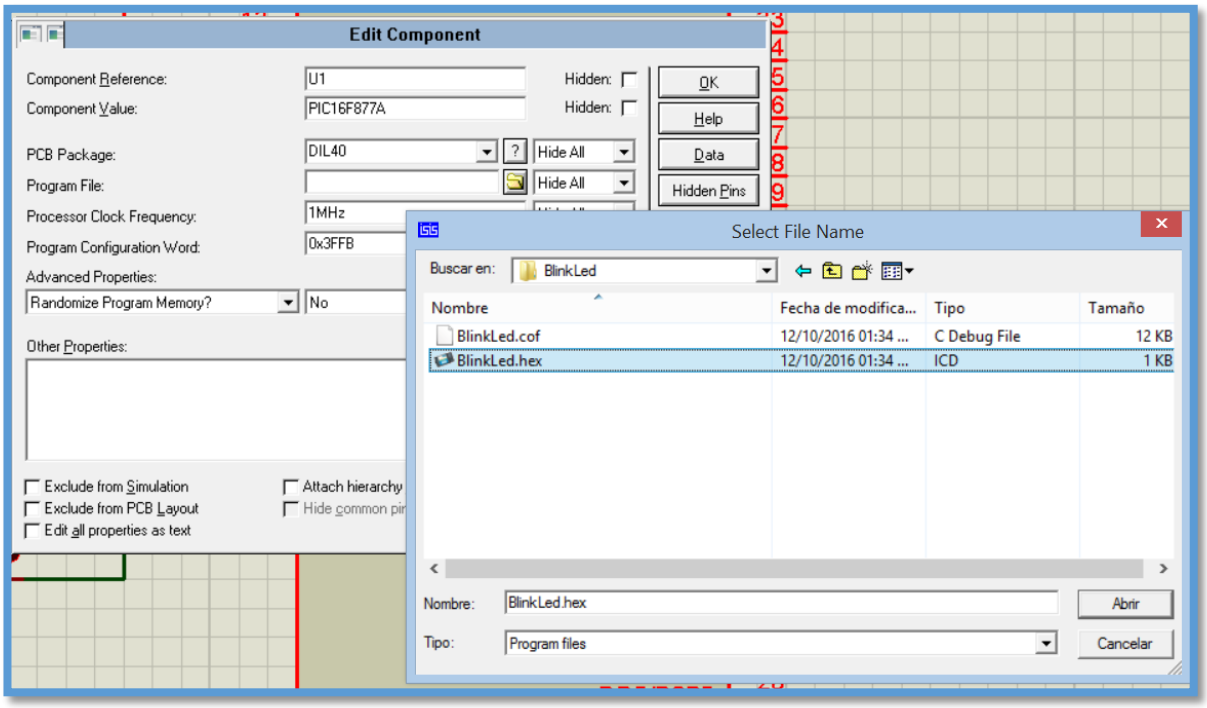
3.2. PASO2

Construcción del circuito eléctrico y conexión de los componentes.

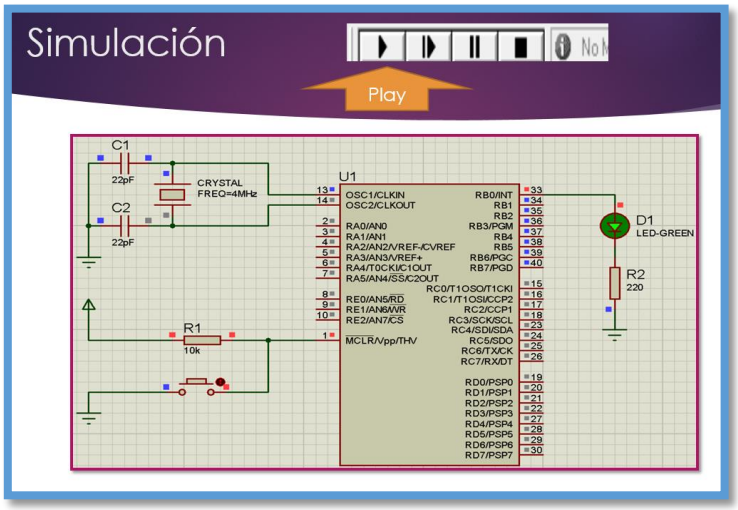


Para realizar las conexiones solo debemos acercar el puntero a la salida de los componentes y lo seleccionamos con un clic luego seleccionamos el extremo de otro componente y se genera una línea verde que representa una conexión eléctrica.

Importar el archivo .HEX para simular el microcontrolador. Para ello nos dirigimos al microcontrolador y le damos doble clic aparecerá la siguiente ventana:



Seleccionamos el icono de PROGRAM FILE e importamos el archivo .HEX. También debemos cambiar la frecuencia de reloj de 1MHz a 4MHz o a la frecuencia que requiera el programador.



3.3. PASO3

4. ESTRUCTURAS DE CONTROL EN LENGUAJE C PARA PROGRAMAR MICROCONTROLADORES

4.1. TIPOS DE DATOS

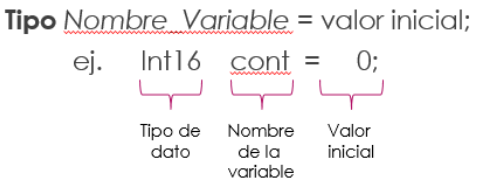
Tipo	Tamaño	Rango	Descripción
Int Int8	8 bit	0 a 255	Entero de 8 bits
Int16 long	16 bit	0 a 65535	Entero de 16 bits
Int32	32 bit	0 a 4294967295	Entero de 32 bits
Float	32 bit		Coma flotante
Char	8 bit	0 a 255	Caracter
Void	-	-	Sin valor
Signed Int8	8 bit	-128 a+127	Entero con signo
Signed Int16	16 bit	-32768 a + 32767	Entero largo con signo
Signed Int32	32 bit	-2^(31) a +(2^(31)-1)	Entero 32 bit con signo

4.2. CONSTANTES

Valor	Descripción
123	Decimal
0123	Octal (0)
0x123	Hexadecimal (0x)
0b00001111	Binario (0b)
'A'	Carácter

4.3. VARIABLES

Variables locales: Las variables globales se pueden usar solo en las funciones donde se crearon.
Variables globales Las variables globales se pueden usar en cualquier parte del programa.
Las variables poseen la siguiente estructura:



4.4. OPERADORES

4.4.1. OPERADORES ARITMETICO

Aritméticos	Descripción
+	Suma
-	Resta
*****	Multiplicación
/	División
%	Resto de una división
--	Decremento en 1
++	Incremento en 1

4.4.2. OPERADORES LOGICOS

Lógicos	Descripción
!	NOT
&&	AND
||	OR

4.4.3. OPERADORES DE ASIGNACION

Asignación	Descripción
+=	Asignación de suma (x+=y es equivalente x = x + y)
-=	Asignación de resta (x-=y es equivalente x = x - y)
***=**	Asignación de multiplicación (x*=y es equivalente x = x * y)
/=	Asignación de división (x/=y es equivalente x = x / y)
%=	Asignación del resto de la división (x%=y es equivalente x = x % y)
<<=	Asignación de desplazamiento a la derecha (x<<=y es equivalente x = x << y)
>>=	Asignación de desplazamiento a la izquierda (x>>=y es equivalente x = x >> y)
&=	Asignación AND de bits (x&=y es equivalente x = x & y)
|=	Asignación OR de bits (x|=y es equivalente x = x | y)
^=	Asignación OR EXCLUSIVA de bits (x^=y es equivalente x = x ^ y)

4.4.4. OPERADORES DE RELACION

Relacionales	Descripción
>	Menor que
<	Mayor que
>=	Mayor igual que
<=	Menor igual que
==	Igual
!=	Distinto
?:	Expresión condicional

4.4.5. OPERADORES DE BITS

Bits	
~	Complemento a 1
&	AND
^	OR EXCLUSIVA
|	OR
>>	Desplazamiento a la derecha
<<	Desplazamiento a la izquierda

4.5. FUNCIONES

4.6. ESTRUCTURAS DE CONTROL

4.6.1. IF-ELSE

```
if (expresion)          -> Primero se evalua la expresion
{                       -> si es cierta (True o 1), se ejecuta
    sentencia_1;        -> la sentencia_1
}
else                    -> en caso contrario (False o 0)
{                       -> se ejecutara
    sentencia_2         -> la sentencia_2
}
```

4.6.2. SWITCH

```
switch(expresion)           -> Primero se evalua la expresion
{                              y en orden al caso adecuado
    case VAL1:                 para ejecutar la sentencia asociada.
        sentencia;          -> si ninguno de los CASE corresponde
        break;                 al VALOR se ejecuta DEFAULT(comando
                               opcional).
    case VAL2:              -> El comando BREAK provoca la salida
        sentencia;             de SWITCH, caso contrario ejecuta
        break;                 el siguiente CASE.

        .....
    default:
        sentencia;
        break;
}
```

4.6.3. FOR

```
for(inicio;cond;incremento)          ->  INICIO:
{                                        Es una variable a la cual se le asigna
    sentencia;                           un valor inicial con el cual controla el bucle
}
                                     ->  COND:
Bucle sin fin:                           Sirva para evaluar ANTES de ejecutar la
                                         sentencia si es cierta o no, en caso ser cierta
for(;;)                                  sale del FOR, caso contrario ejecuta la
{                                        sentencia
    sentencia;
}                                    ->  INCREMENTO O DECREMENTO
                                         Modifica la variable de control despues de
                                         ejecutar el bucle.
```

4.6.4. WHILE

```
while(expresion)          //->    Primero se evalua la expresion si es
{                         //      verdadera ejecuta la sentencia
    sentencia;
}                         //->    Caso contrario (falso) deja de
                          //      ejecutar la sentencia
Bucle sin fin:
while(TRUE)
{
    sentencia;
}
```

4.6.5. DO WHILE

```
do                          ->  Se diferencia del for y while
{                               en la condicion de finalizacion,
    sentencia;                  la sentencia se ejecuta al menos
}while(expresion);              una vez.

Bucle sin fin:              ->  Para salir del bucle se
                                evalua la expresion si es verdadera
do                              seguira ejecutandose caso
{                               contrario salddra del bucle.
    sentencia;
}while(TRUE);
```

5. DESCRIPCION Y MANIPULACION DE FUNCIONES PARA LOS PUERTOS DE ENTRADA Y SALIDA

El compilador ofrece funciones predefinidas para la manipulación de los pines del microcontrolador.

Ahora describiremos las funciones en el siguiente cuadro, la parte sin sombrear son ejemplos:

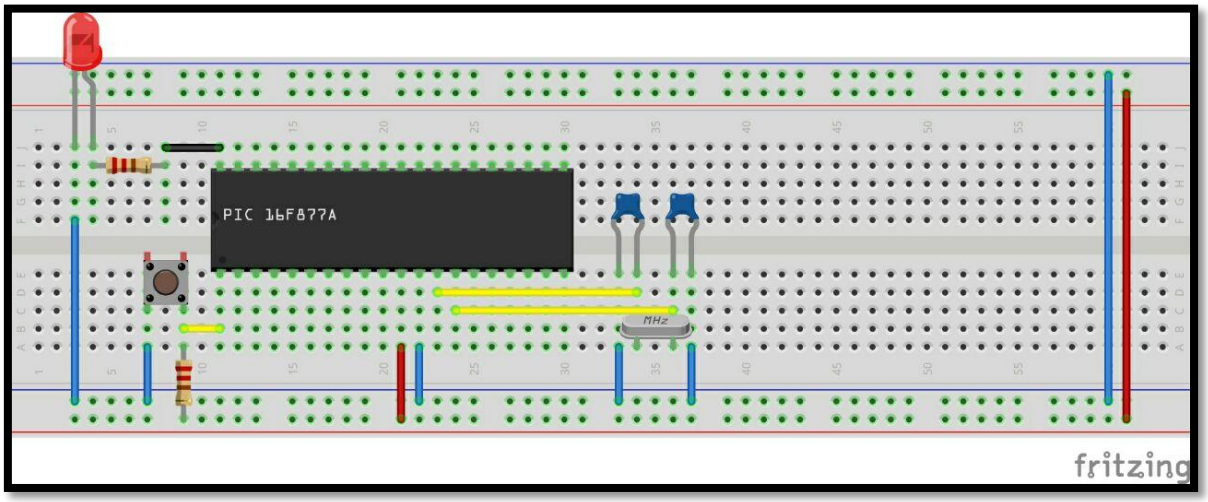
Función	Descripción
SET_TRIS_X(VALOR)	Configura el puerto X como entrada o salida: 1 entrada y 0 salida.
SET_TRIS_B(0B00001111)	Configura el puerto B: B0-B3 entrada y B4-B7 salida.
OUTPUT_X(VALOR)	Saca por el puerto X el valor (0-255).
OUTPUT_C(0B11110000)	Saca por el puerto C: B0-B3 nivel bajo y B4-B7 nivel alto.
PORT_B_PULLUPS(VALOR)	TRUE: Activa resistencias pullups del puerto B. FALSE: Desactiva resistencias pullups del puerto B.
PORT_B_PULLUPS(TRUE)	Activa las resistencias pull up del puerto B. El PIC16f877A posee res. pull up solo en el puerto B.
OUTPUT_LOW(PIN_XX)	El pin XX en nivel bajo.
OUTPUT_LOW(PIN_D0)	El pin D0 en nivel bajo.
OUTPUT_HIGH(PIN_XX)	El pin XX en nivel alto.
OUTPUT_HIGH(PIN_B1)	El pin B1 en nivel alto.
OUTPUT_TOGGLE(PIN_XX)	Complementa el estado del pin XX
OUTPUT_TOGGLE(PIN_C4)	Si C4 se encuentra en nivel alto cambiará a nivel bajo.
VALOR=INPUT(PIN_XX)	Lee el valor del pin XX
VALOR=INPUT(PIN_D0)	Lee el valor de pin D0 previa configuración del TRISX.

6. EJEMPLOS DE LA CLASE1

6.1. EJEMPLO1

Conectar un led al PINB7 y mantenerlo encendido por un tiempo de 500ms y luego apagarlo por un tiempo de 500ms el proceso se repite continuamente.

6.1.1. CONEXIÓN

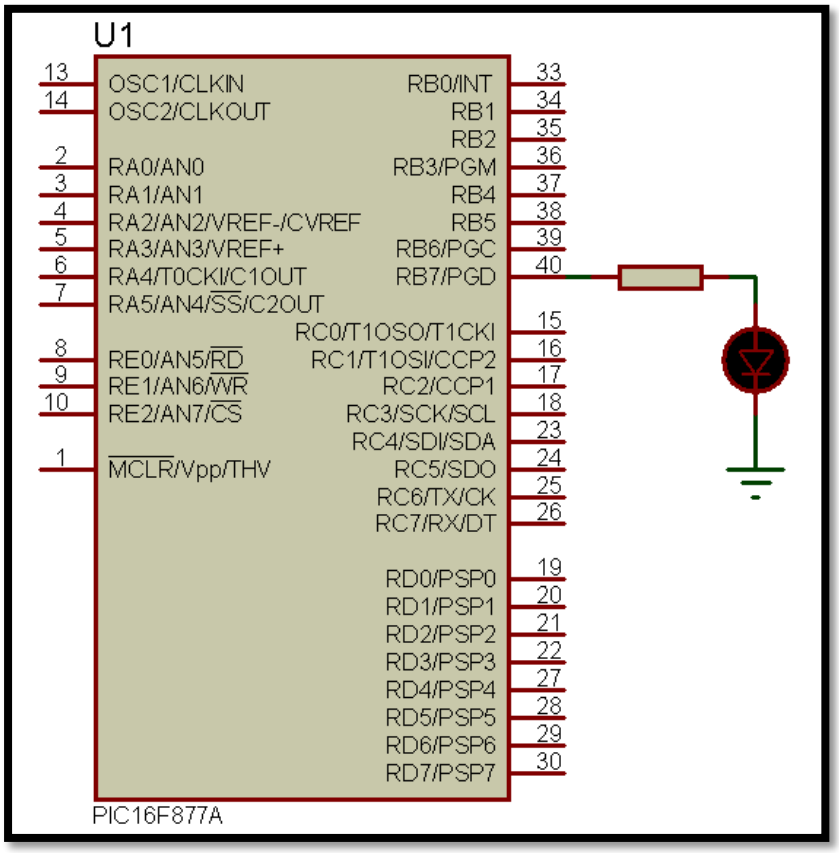


6.1.2. PROGRAMA

ejemplo1.c

```
1  #include <16f877a.h>
2  #use delay(clock=4Mhz)
3
4  void main(void){
5
6     set_tris_b(0x00);
7     output_b(0);
8
9     for(;;){
10       output_high(PIN_B7);
11       delay_ms(500);
12       output_low(PIN_B7);
13    }
14 }
15
```

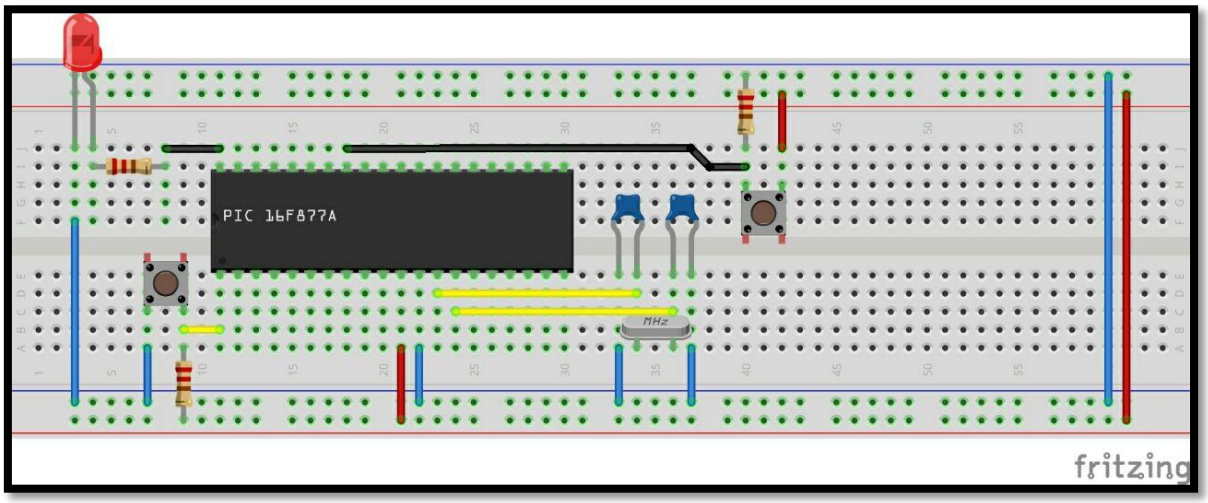
6.1.3. SIMULACION



6.2. EJEMPLO2

Leer el estado de un pulsador conectar al Pin B0 del PIC cuando se presiona el pulsador se enciende un led, cuando no se presione el pulsador se apaga el led, el led debe estar conectado al Pin B7

6.2.1. CONEXIÓN



6.2.2. PROGRAMA

ejemplo2.c

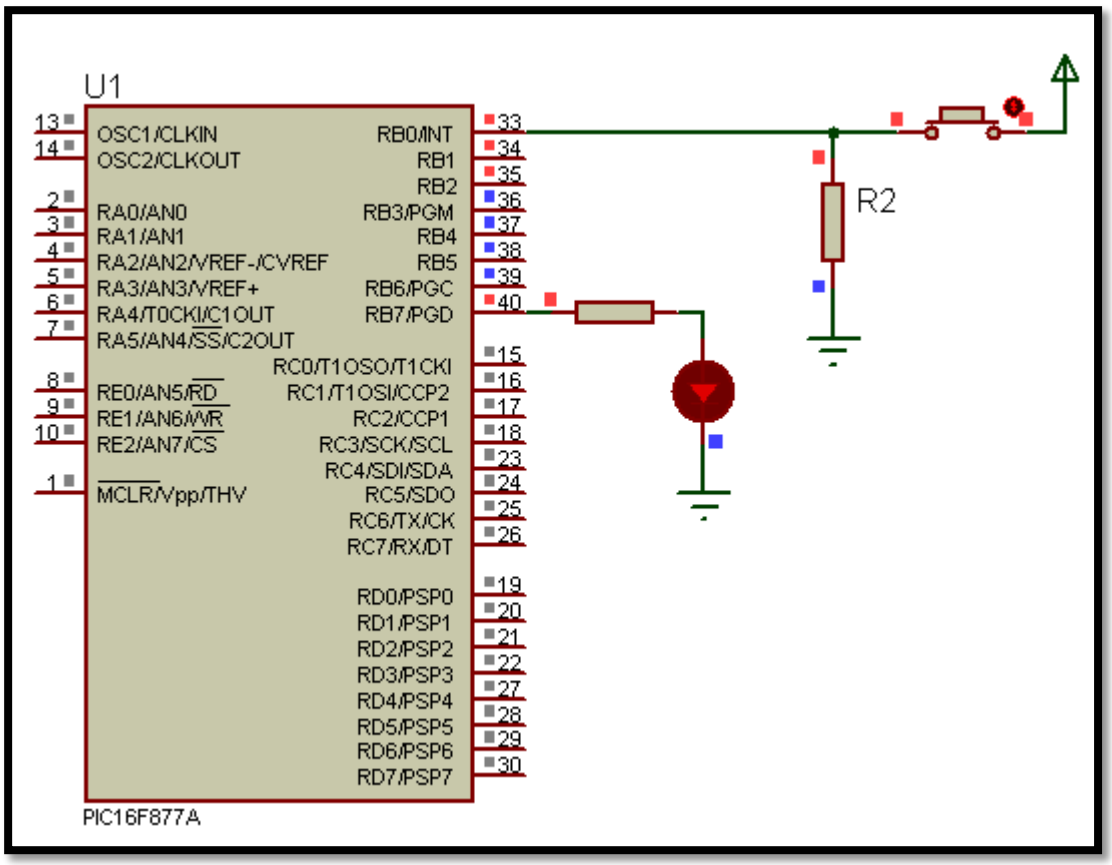
```
#include <16f877a.h>
#fuses XT
#use delay(clock=4Mhz)

void main(void){

   set_tris_b(0b00000001);

   for(;;){
      if(input(PIN_B0)==1){
         output_high( PIN_B7 );
      }else{
         output_low( PIN_B7 );
      }
   }
}
```

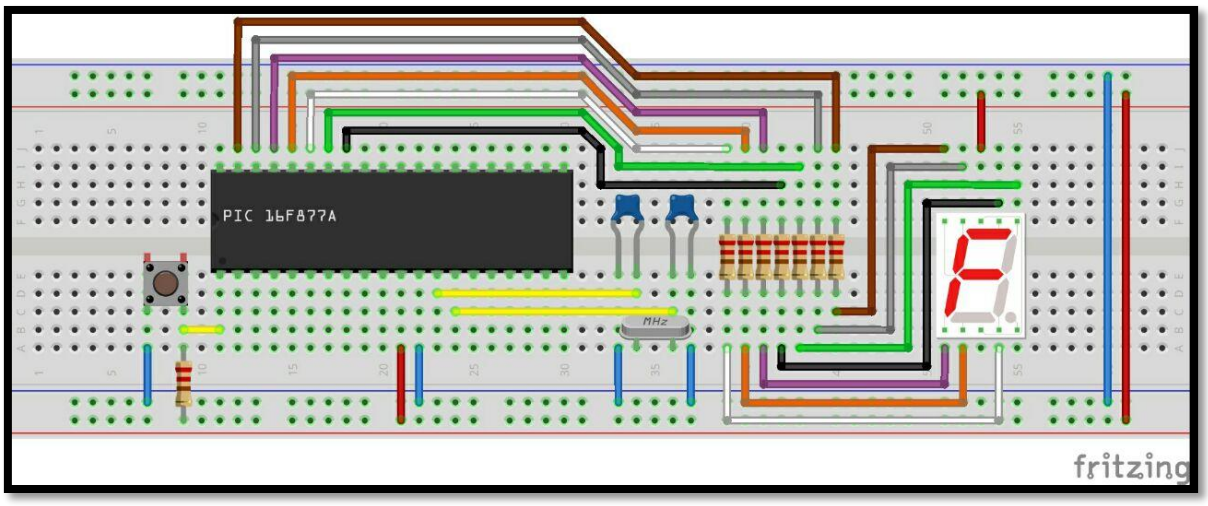
6.2.3. SIMULACION



6.3. EJEMPLO3

Conectar un display de 7 segmentos al puerto B y hacer un conteo de 0 a 9 dicho conteo se muestra en el display continuamente.

6.3.1. CONEXIÓN



6.3.2. PROGRAMA

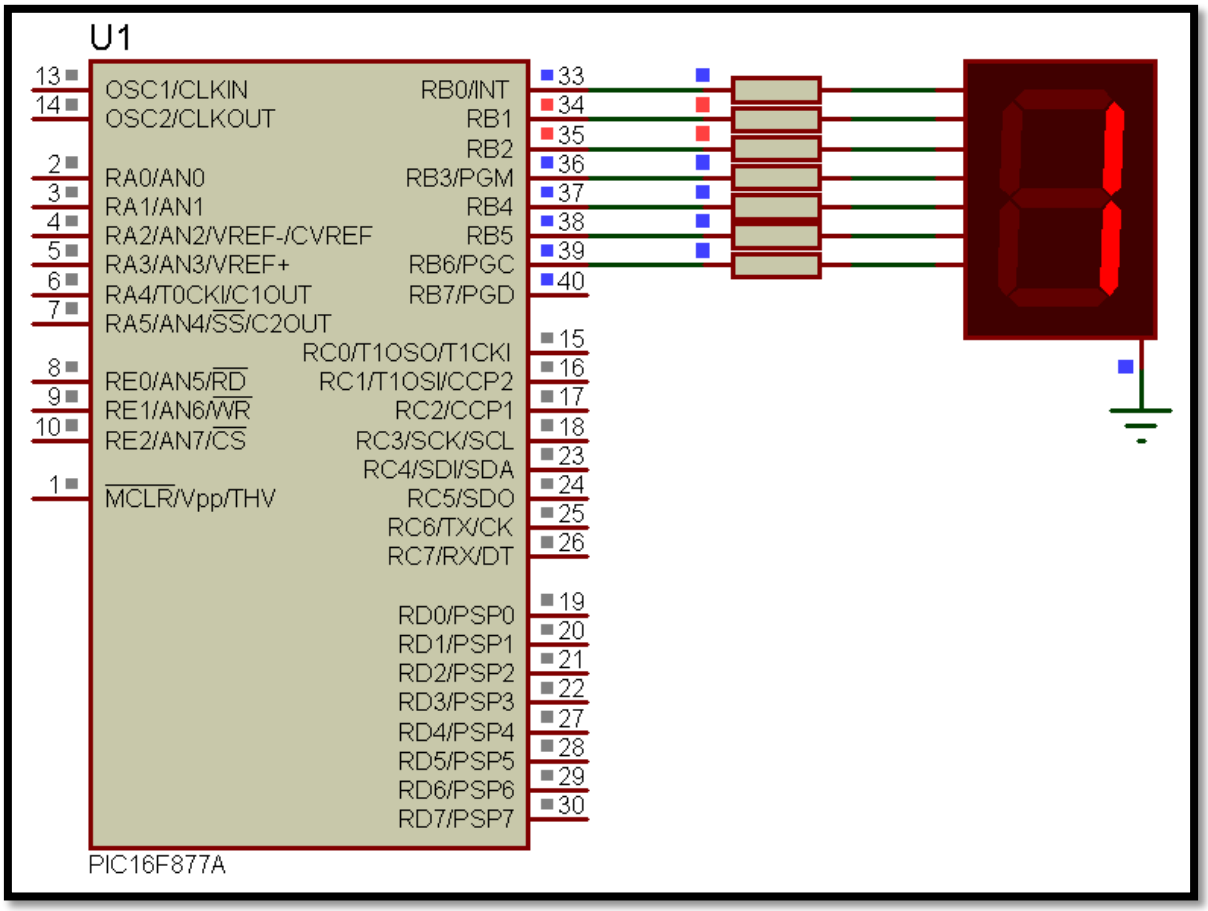
ejemplo3.c

```
#include <16f877a.h>
#fuses XT
#use   delay(clock=4Mhz)
void main(void){

   int valores[10]={ 0x03F,0x06,0x5B,0x4F,0x66,
                     0x6D,0x7D,0x47,0x7F,0x67};
   int i=0;
   set_tris_b(0x00);
   output_b(0);

   for(;;){
      for(i=0;i<10;i++){
         output_b(valores[i]);delay_ms(500);
      }
   }
}
```

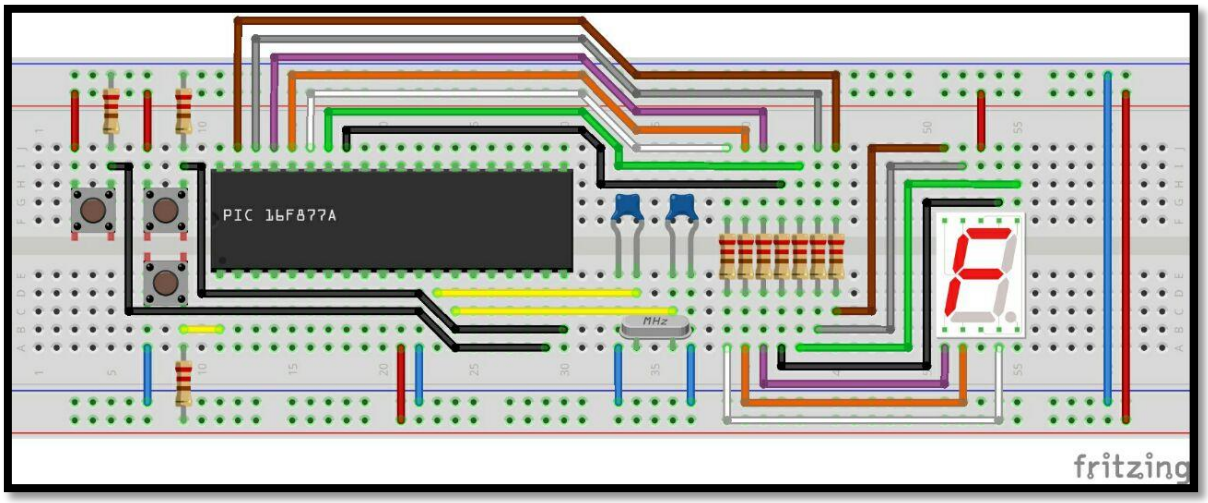
6.3.3. SIMULACION



6.4. EJEMPLO4

Conectar un display de 7 segmentos en el puerto, un pulsador P1 en el pin D1 y un pulsador P2 en el pin D2, cuando presiono P1 inicio la cuenta en el display y cuando presiono P2 se detiene la cuenta

6.4.1. CONEXIÓN



6.4.2. PROGRAMA

ejemplo4.c

```
#include <16f877a.h>
#fuses XT
#use delay(clock=4Mhz)

void main(void){
   int valores[10]={ 0x03F,0x06,0x5B,0x4F,0x66,
                     0x6D,0x7D,0x47,0x7F,0x67   };
   int i=0;
   int bucle=0;
   set_tris_b(0x00);
   set_tris_d(0xFF);
   output_b(0);

   for(;;){
      if( input(PIN_D0) == 1 ){ bucle = 1; }
      if( bucle == 1 ){
         for( i=0; i < 10; i++){
            output_b( valores[i] ); delay_ms( 300 );
            if( input(PIN_D0) == 1 ){
               delay_ms(300);
               bucle = 0;
               break;
            }
         }
      }
   }
}
```

6.4.3. SIMULACION