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 8
Temporizador 0 (Timer 0)

Competencias de La Unidad:

- Emplea las interrupciones en el desarrollo de soluciones a problemas complejos

Resultado de Aprendizaje Propuesto (RAP):

- Usa interrupciones basadas en temporizadores para resolver problemas.

Objetivos de la Práctica:

1. Realizar la simulación de un programa para comprobar su funcionamiento utilizando herramientas computacionales
2. Utilizar el temporizador (Timer 0) en modo contador con la finalidad observar su funcionamiento
3. Implementar técnicas programación con configurar el Timer 0 como contador y temporizador.
4. Implementar programas en un circuito basado en microcontrolador, para comprobar su funcionamiento.

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. Manual de Referencia de CCS

Introducción Teórica

Temporizador (Timer 0)

En algunas ocasiones al utilizar un microcontrolador nos encontramos con la necesidad de contar o generar eventos cada cierto tiempo. Para cumplir con estas tareas, es habitual que los microcontroladores dispongan de circuitos internos para ello. Este circuito, es comúnmente denominado Timer/Counter (Temporizador/Contador) aunque también es habitual encontrarlo con el nombre de RTCC (Real Time Clock Counter).

Con el fin de que se comprenda el funcionamiento y uso del Timer0, se definen los siguientes conceptos:

Frecuencia de oscilación (f_{osc}): Frecuencia de trabajo externa del PIC (un cristal de cuarzo, un resonador, etc.).
Frecuencia interna de instrucciones (f_{int}): Frecuencia del reloj interno de instrucciones generada a partir de la frecuencia de oscilación externa. Para los microcontroladores PIC no coincide con la f_{osc}, siendo una cuarta parte de esta:

$$f_{int}=\frac{f_{osc}}{4}$$

El módulo Timer0 es un temporizador/contador con las siguientes características:

- El temporizador/contador de 8 bits, que puede escribirse y leerse
- Preescaler programable por Software de 8 bits
- Puede trabajar con el reloj interno o con una señal de reloj externa
- Dispone de una interrupción por desbordamiento al pasar de FFh a 00h
- Selección de flanco ascendente o descendente para el flaco del reloj externo

Además, el Timer0 tiene cuatro componentes básicos:

- • La entrada de reloj puede ser desde la patilla RA4/T0CKI o el reloj interno de instrucciones
- • Un circuito divisor de frecuencias programable o prescaler.
- • Un registro contador TMR0.
- • Las banderas de interrupción utilizados por el TIMER0 son: GIE, T0IE y T0IF.

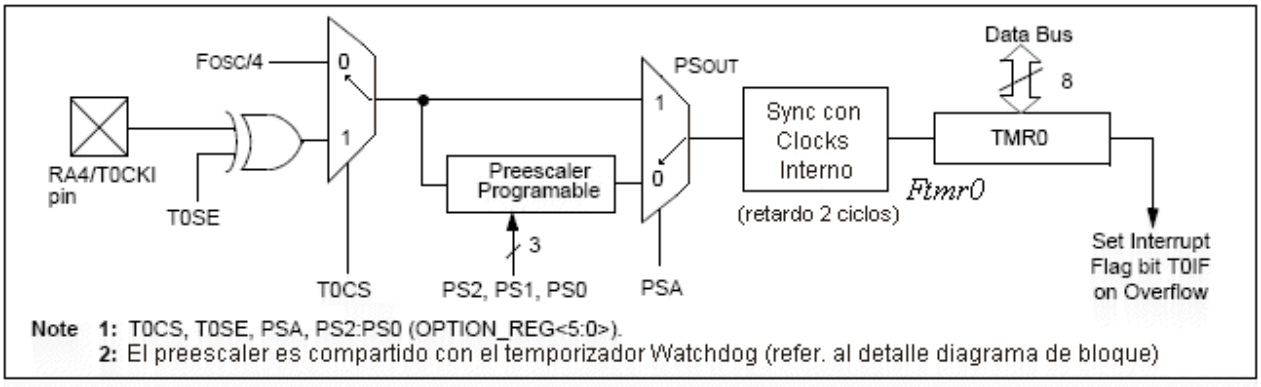

Figura 7.0

Funcionamiento del Timer 0

El Timer0 funciona como un temporizador o contador, según la procedencia de la señal de reloj que recibe. Debemos señalar que en ambos caso funciona de la misma forma, solo que el origen de la señal de entrada cambia y determinara cual de las funciones realizará.

En el caso que señal provenga del reloj interno de instrucciones (***f*$_{int}$**), el Timer0 se utiliza para generar interrupciones periódicas a través de una cuenta programada, pues conocemos la frecuencia de funcionamiento y en base al valor cargado en el contador podemos temporizar tiempos.
Por otra parte cuando la señal se genera en una fuente externa al microcontrolador (***f*$_{ext}$**), es especialmente útil para contar el número de pulsos que dicha señal genera en el tiempo ya que cada pulso de dicha señal incrementa el TMR0.

Entrada de reloj del modulo Timer0

La señal de reloj para el módulo Timer0 se puede obtener de dos formas: a través de la terminal de contador del microcontrolador (RA4/T0CKI), o bien utilizando el reloj interno de instrucciones. En el diagrama mostrado en la figura 7.0 se denominado f_{ext} (RA4/TOCKI) y f_{int} (f_{osc} /4), respectivamente.

Prescaler

Existe un circuito que permite modificar la frecuencia del reloj de entrada del Timer0, dividiendo esta y generando una nueva señal de menor frecuencia a la salida que funcionara como la señal de reloj de entrada al registro TMR0 llamado prescaler, el cual es es activado a través de 4 bits en el registro OPTION, permitiendo dividir la frecuencia de una señal por 1, 2, 4, 8, 16, 32, 64, 128 o 256.

Por ejemplo, si se utiliza un divisor por 1, la señal de salida es la de entrada sin ningún cambio, es decir, si se utiliza un oscilador externo de 4Mhz, entonces el reloj interno de instrucciones funciona a f_{int} = 4Mhz /4 = 1 Mhz, pero en el caso de configurar el prescaler para una división por 4, la señal de salida en el prescaler sera de f_{presc} = 250 Khz. Cabe mencionar, que no se debe confundir la frecuencia de trabajo del modulo Timer0, con la frecuencia de trabajo del registro TMR0 del Timer0: la primera es la frecuencia base utilizada por el modulo, mientras que la segunda es la frecuencia modificada que alimenta al registro TMR0.

El registro TMR0

El registro TMR0 es el elemento primordial del módulo Timer0. Es un registro contador de 8 bits cuyo valor se incrementa automáticamente en cada ciclo de su señal de reloj f_{tmr0} que es la frecuencia interna de trabajo del Timer0 (que puede ser f_{ext} o f_{int}).

La finalización de la cuenta se detecta cuando el contador pasa por 0: es decir, cuando el valor del registro TMR0 pasa de 255 a 0 (0xFF a 0x00), activando la bandera T0IF indicando que ha ocurrió un desbordamiento (“overflow”) y el registro continua incrementándose normalmente con cada pulso del prescaler.

Cálculo de temporizaciones

Supongamos que el módulo Timer0 se alimenta con una señal de frecuencia $f_{timer0} = f_{osc}/4 = f_{int}$. Por lo tanto el tiempo de desbordamiento del Timer 0 (T_{retmr0}) se calcula de la siguiente manera:

$$T_{retmr0} = T_{CM} * Prescaler * (256 - Valor\ cargado\ en\ TMR0)$$

Donde: T_{CM} es el ciclo de máquina que equivale a $1/f_{int} = 4/f_{osc}$

ACTIVIDADES TEÓRICAS PREVIAS

Investigar los siguientes:

- ***Investigar que son interrupciones***
- ***¿Qué tipo de interrucciones tienen los Microcontroladores PIC?***
- **Diagrama interno del Timer 0**
- **Instrucciones para configurar la interrupciones globales en lenguaje C**
- **Instrucción para inicializar el valor del Timer 0 y configurarlo en lenguaje C**
- **Que características tiene el Prescaler**
- **¿Cuáles son las banderas afectas por el registro del Timer 0?**

ACTIVIDADES PREVIAS

- ***Crear un proyecto de nombre pra8 en la carpeta c:\MEPIC\practica8 en MPLAB o PIC C Compiler. Los programas de cada ejercicio deben ser guardados con el nombre practica8X.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.***

ACTIVIDADES PRÁCTICAS
Parte 1

1. Armar el siguiente circuito

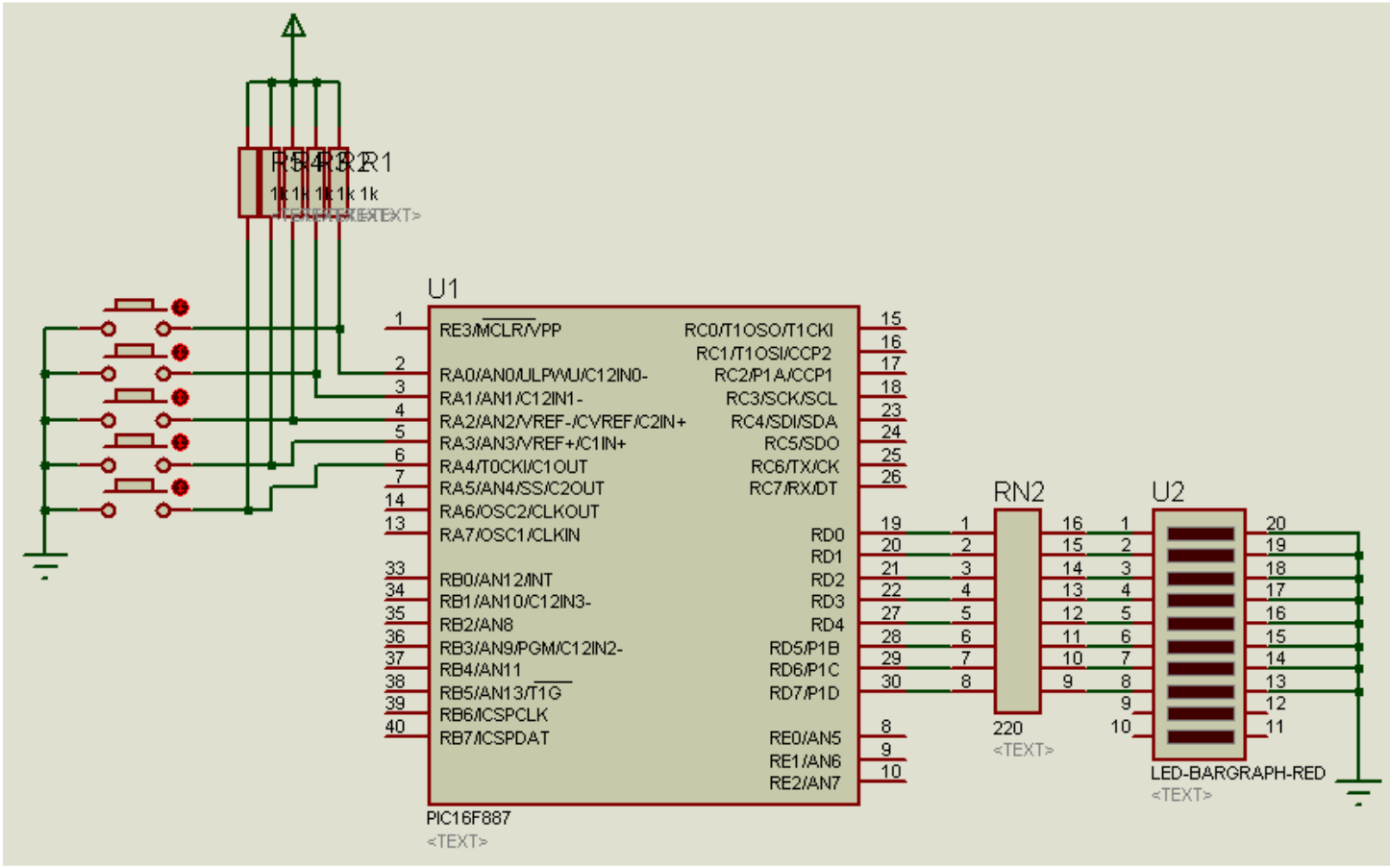

Figura 7.1

Notas: La terminal 11 o 32 del PIC16F887 se conectan a tierra.
A grabar el microcontrolador deshabilitar WDT y LVP y habilitar PWRTE y BODEN en la palabra de configuración, además recuerde seleccionar el tipo de oscilador a XT

Ejemplo 1
El siguiente programa, utiliza el timer 0 en modo contador, detectado y contándolo los cambios de nivel presentados en la terminal RA4/T0CKI, y al llegar la cuenta 10 prende un led conectado a RB7, arme el circuito de la figura 7.1, grabarlo en el microcontrolador, así como simularlo en PROTEUS.

```
#include <16f887.h>
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use delay(clock=4000000)
int datos[11] = {0x0A,0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01,0x00};
int timer0;
int i=11;
#int_TIMER0
void TIMER0_isr(void) {
```

```
i=0;
output_b(datos[i]);
output_high(PIN_d7);
delay_ms(1000);
output_low(PIN_d7);
set_timer0(246); //se recarga el timer0
output_b(datos[i]);
}
void main()
{
setup_timer_0(RTCC_EXT_H_TO_L|RTCC_DIV_1); //Configuración timer0
enable_interrupts(INT_TIMER0); //Habilita interrupción timer0
enable_interrupts(global); //Habilita interrupción general
set_timer0 (246); //Carga del timer0
output_d(datos[i]);
do{
timer0=get_timer0();
i=~timer0+1;
output_d(datos[i]);}
while(true);
}
```

Ejemplo 2

Realizar el siguiente programa, que permita mostrar los valores 11110000 y 0000111 de manera intercalada cada 0.5 segundos.

- ***Grabarlo circuito de la figura 7.1***
- ***Simularlo en PROTEUS***
- ***Utilizando stopwatch de MPLAB verificar el tiempo que tarda la interrupción.***

```
#include <16F887.h>
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use delay(clock=4000000) //frecuencia
#use standard_io(d)
//variables globales
int nint=250;
int salida=0xf0;
#INT_TIMER0
void TIMER0_isr(void)
{
set_timer0 (256-250);
--nint;
if(nint==0)
{
salida=~salida;
output_d(salida);
nint=250;
}
}
void main(){
output_d(salida);
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_8); //Configuración timer0
set_timer0 (6); //Carga del timer0
enable_interrupts(INT_TIMER0); //Habilita interrupción timer0
enable_interrupts(global);
while (true) {
}
}
```

Ejemplo 3

El siguiente programa genera una señal de una frecuencia de un 1Khz en el puerto RD0 con un ciclo de trabajo de 50% y se representa mediante el encendido y apagado led. Observe la señal en el simulador mediante el osciloscopio (únicamente simule en Proteus y proponga el circuito).

Es importante recordar que 1Khz es igual a 1000 µs, en este caso como se requiere un ciclo de trabajo de 50 % es decir 500µs, por tanto se deben de contar 500 ciclos de máquina, por tal motivo se ha uso del preescaler para incrementar el registro del timer cada 2 ciclos, es decir se usara un prescaler de 2, el cual al contar 250 sucesos se obtendrá los 500 ciclos de máquina. En este caso el timer se inicializa en 6.

```
#include <16F887.h>
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use delay(clock=4000000) //frecuencia
#use standard_io(d)

//variables globales

int salida=0x00;
#INT_TIMER0
void TIMER0_isr(void)
{

salida=(~salida&0x01);
output_d(salida); // son equivalentes a
//output_toggle(PIN_B0);
// En estas instrucciones
//permite habilitar salida individuales
set_timer0 (6);

}
void main()
{
output_d(salida);
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2); //Configuración timer0
set_timer0 (6); //Carga del timer0
enable_interrupts(INT_TIMER0); //Habilita interrupción timer0
enable_interrupts(global);
while (true)
{
}
}
```

Nota: En lenguaje C la temporización de frecuencias altas tiene un problema de exactitud debido a que no es posible ajustar los ciclos de maquina como en ensamblador. Pero se puede lograr temporizaciones adecuados en función de la aplicación.

Ejemplo 4

Realizar el siguiente programa y simularlo en PROTEUS , utilizando el circuito de la figura 7.2
El programa lee el valor de la entrada conectada a RA0 cada 20ms, el cual es 1 si la el valor leído por el sensor LM35 es mayor a 25° y cero en temperaturas menores.

- ***Simularlo en PROTEUS***
- ***Utilizando stopwatch de MPLAB verificar el tiempo que tarda la interrupción.***

```
#include <16F887.h>
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
#use delay (clock=4M) //frecuencia
#use standard_io(a)
#use standard_io(d)
int a;
#INT_TIMER0
void TIMER0_isr(void)
{
a=input_a()&1;
set_timer0 (170);
}
void main(){
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256); //Configuración timer0
set_timer0 (170); //Carga del timer0
enable_interrupts(INT_TIMER0); //Habilita interrupción timer0
enable_interrupts(global);
output_d(0b00001001);
a=input_a()&1;
while (true) {
if(a==1)
output_d(0b00000100);
else
{ output_d(0b00001001); }
}
}
```

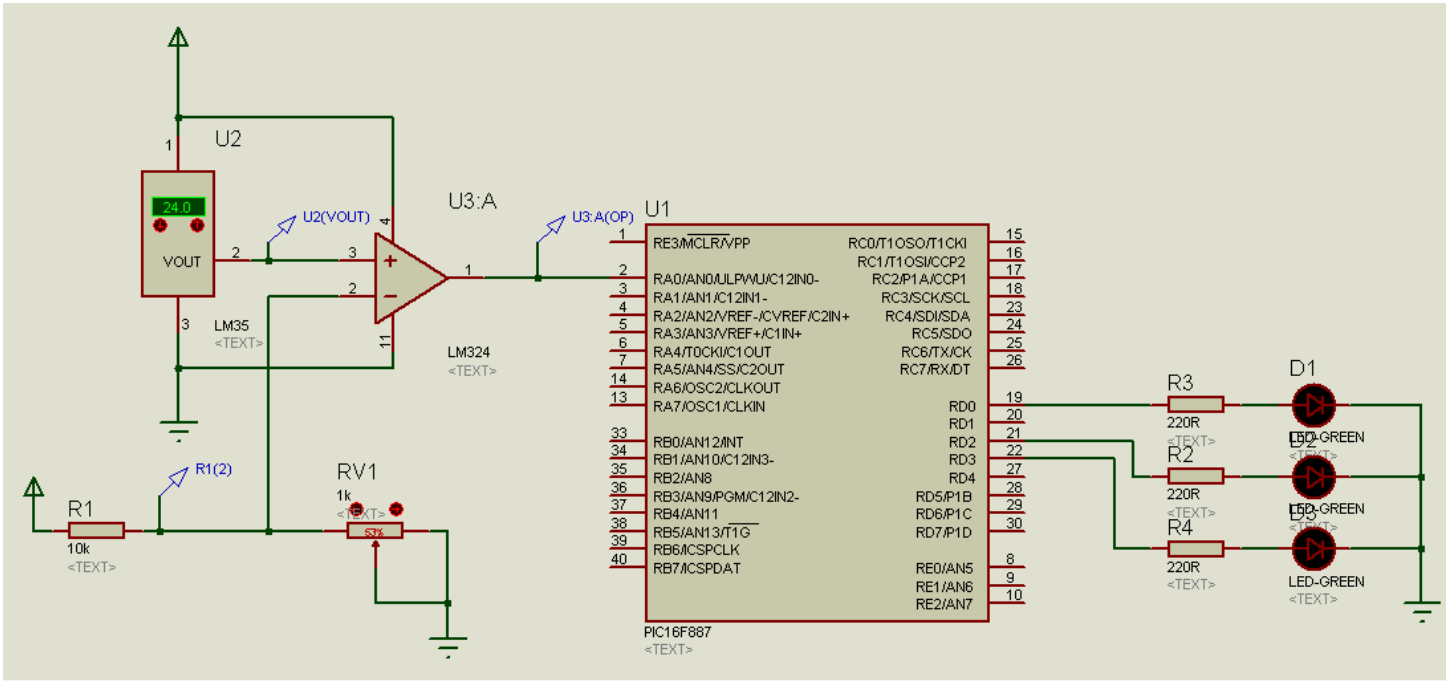

Figura 7.2

Ejercicio 5 Realizar el siguiente programa y unicamente simularlo en PROTEUS , utilizando el circuito de la figura 7.2

El programa efectúa la misma tarea que el ejemplo 4 pero en este caso verifica las interrupción mediante el uso de la bandera de interrupción del Timer0

- ***Simularlo en PROTEUS***
- ***Utilizando stopwatch de MPLAB verificar el tiempo que tarda la interrupción.***

```
   #include <16F887.h>
#fuses XT,NOWDT,NOPUT,NOMCLR,NOPROTECT,NOCPD,NOBROWNOUT,NOIESO,NOFCMEN,NOLVP
   #use delay (clock=4M) //frecuencia
   #use standard_io(a)
   #use standard_io(d)
   int a;
   #byte INTCON= 0x0B
   void main(){
   //INTCON=0X00; es similar a deshabilitar la interrupcion
   bit_clear(INTCON,2); //limpia bandera de interrupcion de TIMER0
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256); //Configuración timer0
   //Carga del timer0
   disable_interrupts(INT_TIMER0); //deshabilita interrupción timer0
   disable_interrupts(global);
   output_b(0b00001001);
   while (true) {
   bit_clear(INTCON,2);
   set_timer0 (170);
   a=input_a()&1;
   if(a==1)
   output_d(0b00000100);
   else
   { output_d(0b00001001); }
   while(bit_test(INTCON,2)!=1)
   {
```

```
}
delay_us(1);
}
}
```

Parte 2

Realizar los siguientes programas y sus respectivas simulaciones, así como la implementación en el circuito armado en el caso de indicarse.

A. Realice un programa de que cuente los pulsos detectados en RA4/T0CK1, el número de pulsos debe visualizar en un LCD además de "Vacio" y al llegar 150 pulsos active un led conectado a un puerto y se visualice "lleno" durante 3 segundos.

B. Realizar un programa mantenga encendido y apagado durante 2 segundos (alternadamente) un motor de cd, además de y tener un indicar de encendido en RB6 y de apagado en RB2. Proponga un circuito que solución el problema y realice la simulación en Proteus

C. Modifique el ejemplo 4 para que la lectura de la entrada realice cada 40ms:
Realice la simulación en Proteus

D. Modifique el ejemplo 5 para que la lectura de la entrada realice cada 4ms:
Realice la simulación en Proteus

2. Conclusiones

A. Realizar conclusiones de manera individual.

3. Cuestionario

a) ¿Exprese una ecuación para determinar el valor que tiene que se colocado en registro TMR0?
b) ¿Qué prescaler utilizaría para obtener temporización de 2ms?
c) ¿Menciona los pasos para configurar el timer0 en C?
d) ¿Qué registros están asociado al Timer 0?
e) ¿En el ejemplo 4 cada cuanto tiempo se lee la entrada?
f) ¿Qué valores de prescaler se pueden tener?
g) ¿Qué procedimiento es necesaria realizar si requiere leer la bandera de interrupción del TIMER0?
h) ¿Cuál es valor máximo de temporización con un preescaler de 256?
i) ¿Cuál es valor máximo de temporización con los preescaler de 2,4,8?
j) ¿Cómo logro realizar temporización mayores a 65ms?

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.**