**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FIEC**

****

**Laboratorio de Microcontroladores**

**Proyecto#1**

**TEMPORIZACIÓN CON EL TMR0**

**Nombre:**

**Israel Ruiz Quinde.**

**Paralelo: # 4**

**Grupo: # 1**

**Fecha de presentación:**

**11/07/2011**

**2011 – I TÉRMINO**

**PROYECTO PRIMER PARCIAL**

**Temporización con el TMR0**

1. **ENUNCIADO DEL PROYECTO:**

Se trata de un programa que nos permite usar un microcontrolador para implementar retardos de tiempo usando la fórmula de temporización con el TMR0 para hacer parpadear un LED ROJO por un minuto con retardos de 15ms. Luego hacer parpadear un LED VERDE por 1 minuto con retardos de 30ms.

En su primera etapa este proyecto será realizado en su parte lógica es decir se presenta en esta carpeta y en CD adjunto el listado del programa, el código en assembler y la simulación en Proteus.

1. **DIAGRAMA DE BLOQUES**

Microcontrolador

PIC: 16F887

**RB1 RB2**

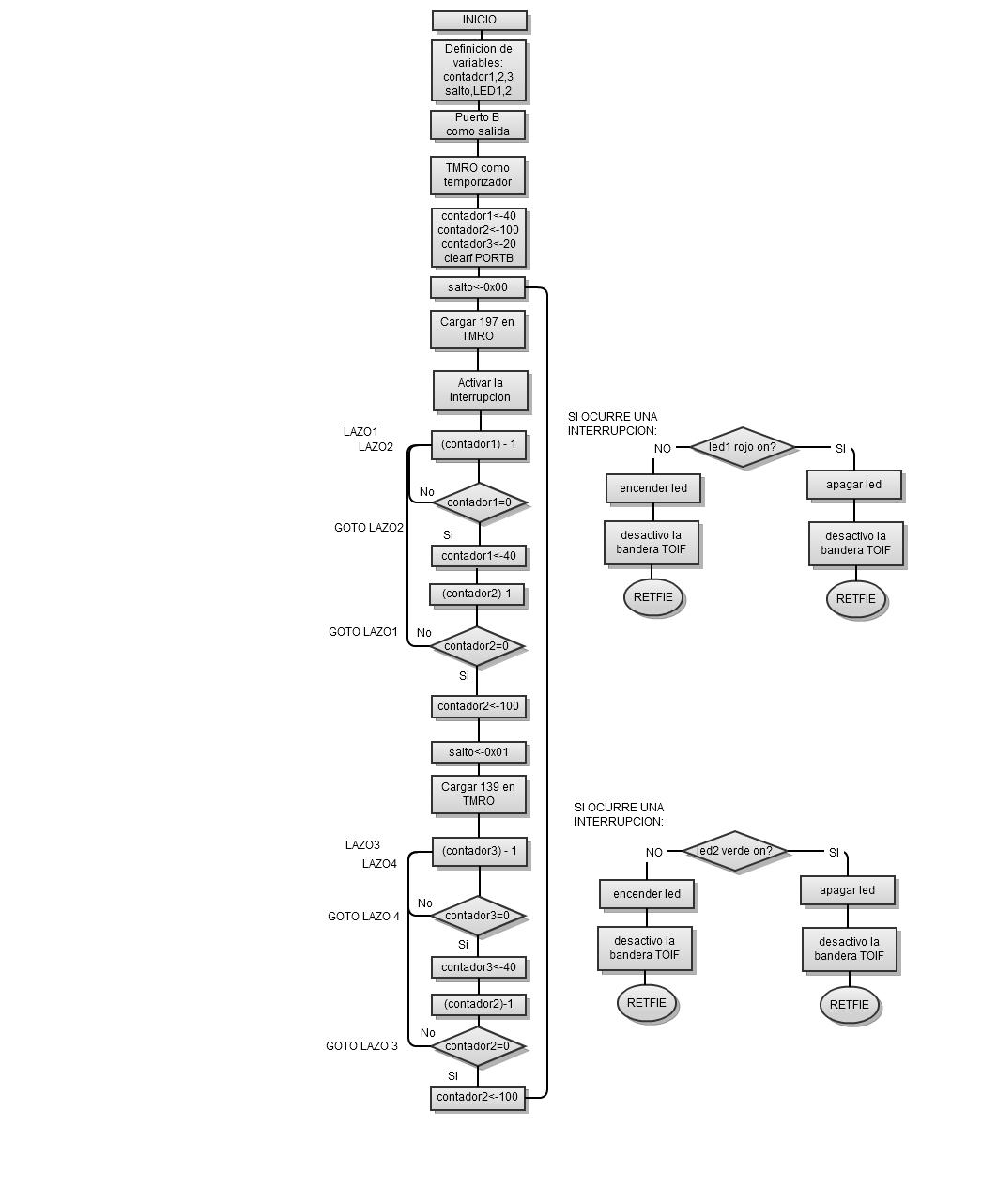
LED2

Verde

LED1

Rojo

1. **DIAGRAMA DE FLUJO FUNCIONAL DEL PRIGRAMA PRINCIPAL:**

****

1. **DIAGRAMA DE FLUJO DE CADA UNA DE LAS SURUTINAS EMPLEADAS**

No se utilizaron subrutinas en este programa.

1. **ALGORITMO:**
2. Comienzo del programa
3. Se escoge el pic p16f887.
4. Se configuran los parámetros:
   * + CP desactivado
     + WDT deshabilitado
     + Se elige el oscilador XT
     + Se hace uso de errorlevel -302
5. Se definen y se inicializan las variables a utilizar:

salto ;si es 0x00 voy a interrupcion1 sino voy a ineterrupcion2

contador1 ; Cuenta 40 interrupciones de 15ms

contador2 ; Cuenta 100 veces

contador3 ; Cuenta 20 interrupciones de 30ms

#DEFINE LED1 PORTB,1 ;Escojo solo el bit 1 del PuertoB(RB1)

#DEFINE LED2 PORTB,2 ;Escojo solo el bit 2 del PuertoB(RB2)

1. Configuramos los registros ANSEL, ANSELH, TRISB en este caso para tener todas los pines de PORTB como salidas digitales.
2. Se inicializan en cero los bits de los puertos de salida para mantener los apagados.
3. Programar TMR0 como contador con pre escalador 256, con los bits del registro OPTION.
4. Se inicializan las variables. Contador1 con 40 y contador2 con 100 contador3 con 20
5. Cargar el TMR0 con 197 para obtener una interrupción cada 15mseg, y activamos dicha interrupción con los bits del registro INTCON.
6. Se decrementa contador1 y cuando ocurra una interrupción se deshabilita la bandera TOIF del registro INTCON para continuar con la siguiente interrupción.
7. Mientras que contador1sea distinto de cero, se recarga 197 en TMR0. En la primera interrupción se enciende el led rojo, en la siguiente se lo apaga y así sucesivamente.
8. Cuando contador1 llegue a cero entonces se lo vuelve a cargar con su valor inicial y se decrementa el contador2. Se repite el paso 11) hasta que contador2 llegue a cero. Con esto se habrá conseguido el primer minuto (100x40x15ms) = 60000ms = 60s
9. Si contador2 llegó cero se lo recarga con su valor inicial y se carga el TMRO esta vez con 139 para obtener una interrupción cada 30mseg.
10. Se decrementa el contador3 y cuando ocurra una interrupción se deshabilita la bandera TOIF para continuar con la siguiente interrupción.
11. Mientras que contador3sea distinto de cero, se recarga 139 en TMR0. Esta vez en la primera interrupción se enciende el led verde, en la siguiente se lo apaga y así sucesivamente.
12. Cuando contador3 llegue a cero entonces se lo vuelve a cargar con su valor inicial y se decrementa el contador2. Se repite el paso 11) hasta que contador2 llegue a cero. Con esto se habrá conseguido el segundo minuto (100x20x30ms) = 60000ms = 60s
13. Si contador2 llegó a cero se lo recarga con su valor inicial y se repite el proceso para encender el led1 rojo y luego para encender el led2 rojo y esto se queda ciclado.
14. **LISTADO DEL PROGRAMA**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; CONTADOR CON DISPLAY DE SIETE SEGMENTOS**

**; USANDO EL TMR0 COMO BASE DE TIEMPO DE 10ms**

**; PARA CONTAR CADA SEGUNDO**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; NOMBRE: proyecto1.asm**

**; FECHA: 11/07/2011**

**; Autor: Israel Ruiz**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**; DESCRIPCION:**

**;Usando la formula de temporización con el TMRO**

**;hacer parpadear un led rojo con retardos de 15ms por un minuto**

**;luego hacer paradear un led verde con retardos de 30ms por un minuto**

**;Use reloj de 4MHz.**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**;DIRECTIVAS**

**LIST p=16F887 ;Tipo de microcontrolador**

**INCLUDE P16F887.INC ;Define los SFRs y bits del**

**;P16F887**

**\_\_CONFIG \_CONFIG1, \_CP\_OFF&\_WDT\_OFF&\_XT\_OSC**

**;Setea parámetros de**

**;configuración**

**errorlevel -302 ;Deshabilita mensajes de**

**;advertencia por cambio bancos**

**;La directiva "cblock 0X20...endc" reserva espacios de memoria a partir de 0x20**

**;INICIO DE PROGRAMA**

**cblock 0x20**

**salto ;si es 0x00 voy a interrupcion1 sino voy a ineterrupcion2**

**contador1 ;Cuenta 40 interrupciones de 15ms**

**contador2 ;Cuenta 100 veces**

**contador3 ;Cuenta 20 interrupciones de 30ms**

**Endc**

**#DEFINE LED1 PORTB,1 ;Escojo solo el bit 1 del PuertoB(RB1)**

**#DEFINE LED2 PORTB,2 ;Escojo solo el bit 2 del PuertoB(RB2)**

**ORG 0x00 ;Vector de RESET**

**GOTO MAIN**

**ORG 0x04 ;Vector de interrupción**

**btfss salto,0 ;Escojo entre interrupcion1 e interrupcion2**

**GOTO INTERRUPCION1 ; se enciende el led verde**

**GOTO INTERRUPCION2 ; se enciende el led rojo**

**INTERRUPCION1**

**decf contador1,f ;Decrementa el contador1 40 veces**

**btfsc LED1 ; comprueba el estado del LED1 encendido o apagado**

**GOTO EstabaEncendido**

**EstabaApagado ; si estaba apagado lo enciende**

**bsf LED1**

**GOTO Seguir**

**EstabaEncendido ; si estaba encendido lo apaga**

**bcf LED1**

**Seguir**

**movlw .197**

**movwf TMR0 ;Repone el TMR0 con 197**

**bcf INTCON,T0IF ;Repone flag del TMR0**

**RETFIE ;Retorno de interrupción**

**INTERRUPCION2**

**decf contador3,f ;Decrementa el contador3 20 veces**

**btfsc LED2 ;comprueba el estado del LED2 encendido o apagado**

**GOTO EstabaEncendido2**

**EstabaApagado2 ;si estaba apagado lo enciende**

**bsf LED2**

**GOTO Seguir2**

**EstabaEncendido2 ;si estaba encendido lo apaga**

**bcf LED2**

**Seguir2**

**movlw .139**

**movwf TMR0 ;Repone el TMR0 con 197**

**bcf INTCON,T0IF ;Repone flag del TMR0**

**RETFIE ;Retorno de interrupción**

**MAIN**

**;SETEO DE PUERTOS**

**BANKSEL ANSEL**

**CLRF ANSEL**

**CLRF ANSELH**

**BANKSEL TRISB ;Selecciona el Bank1**

**CLRF TRISB ;PORTB configurado como salida**

**;PROGRAMACION DEL TMR0**

**banksel OPTION\_REG**

**movlw b'00000111' ;Programa TMR0 como temporizador**

**movwf OPTION\_REG ;con preescaler de 256**

**;INICIALIZACION**

**BANKSEL PORTB ;Selecciona el Bank 0**

**CLRF PORTB ;borra el contenido de PORTB**

**CLRF salto**

**movlw .40 ;Cantidad de interrupciones a contar**

**movwf contador1 ;Nº de veces a repetir la interrupción**

**movlw .100 ;Cantidad de interrupciones a contar**

**movwf contador2 ;Nº de veces a repetir la interrupción**

**;El valor calculado para cargar en TMR0 es de 197 con un preescaler**

**;de 256 y a una frecuencia de 4MHz, para obtener una interrupción**

**;cada 15mS.**

**movlw .197 ;Valor decimal 197**

**movwf TMR0 ;Carga el TMR0 con 197**

**;PROGRAMACION DE INTERUPCION**

**movlw b'10100000'**

**movwf INTCON ;Activa la interrupción del TMR0**

**Loop1**

**Loop2**

**movf contador1,0**

**xorlw b'00000000' ;contador1 ha llegado a cero?**

**btfss STATUS,Z ;si Z=1 ejecuto el salto**

**GOTO Loop2 ; aun no**

**movlw .40**

**movwf contador1 ;dejo cargado contador1 con su valor inicial**

**decfsz contador2,f ;decremento contador2 100 veces y ejecuto salto si este llega a cero**

**GOTO Loop1**

**bcf STATUS,Z ;desactivo flag Z**

**bsf salto,0 ;me permite seleccionar la interrupcion2**

**movlw .100**

**movwf contador2 ;dejo cargado contador2 con su valor inicial**

**movlw .139 ;Valor decimal 139**

**movwf TMR0 ;Carga el TMR0 con 139**

**movlw .20 ;Cantidad de interrupciones a contar**

**movwf contador3 ;Nº de veces a repetir la interrupción2**

**;PROGRAMACION DE INTERUPCION**

**movlw b'10100000'**

**movwf INTCON ;Activa la interrupción del TMR0**

**Loop3**

**Loop4**

**movf contador3,0**

**xorlw b'00000000' ;contador3 a llegado a cero?**

**btfss STATUS,Z ;si Z=1 ejecuto el salto**

**GOTO Loop4 ;aun no**

**movlw .20**

**movwf contador3 ;dejo cargado contador3 con su valor inicial**

**decfsz contador2,f ;decremento contador2 100 veces y ejecuto salto si este llega a cero**

**GOTO Loop3**

**bcf STATUS,Z ;desactivo flag Z**

**bcf salto,0 ;me permite volver a la interrupcion1**

**movlw .100**

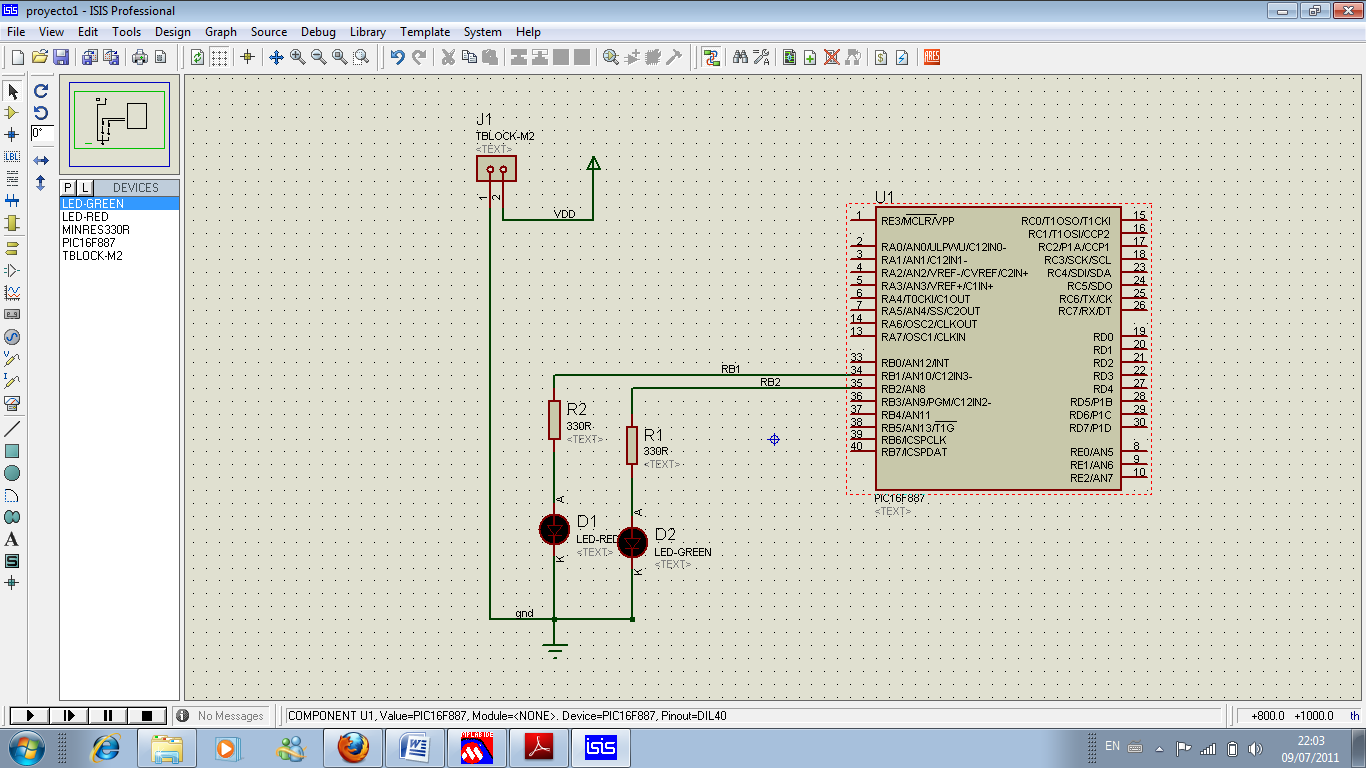
**movwf contador2 ;dejo cargado contador2 con su valor inicial**

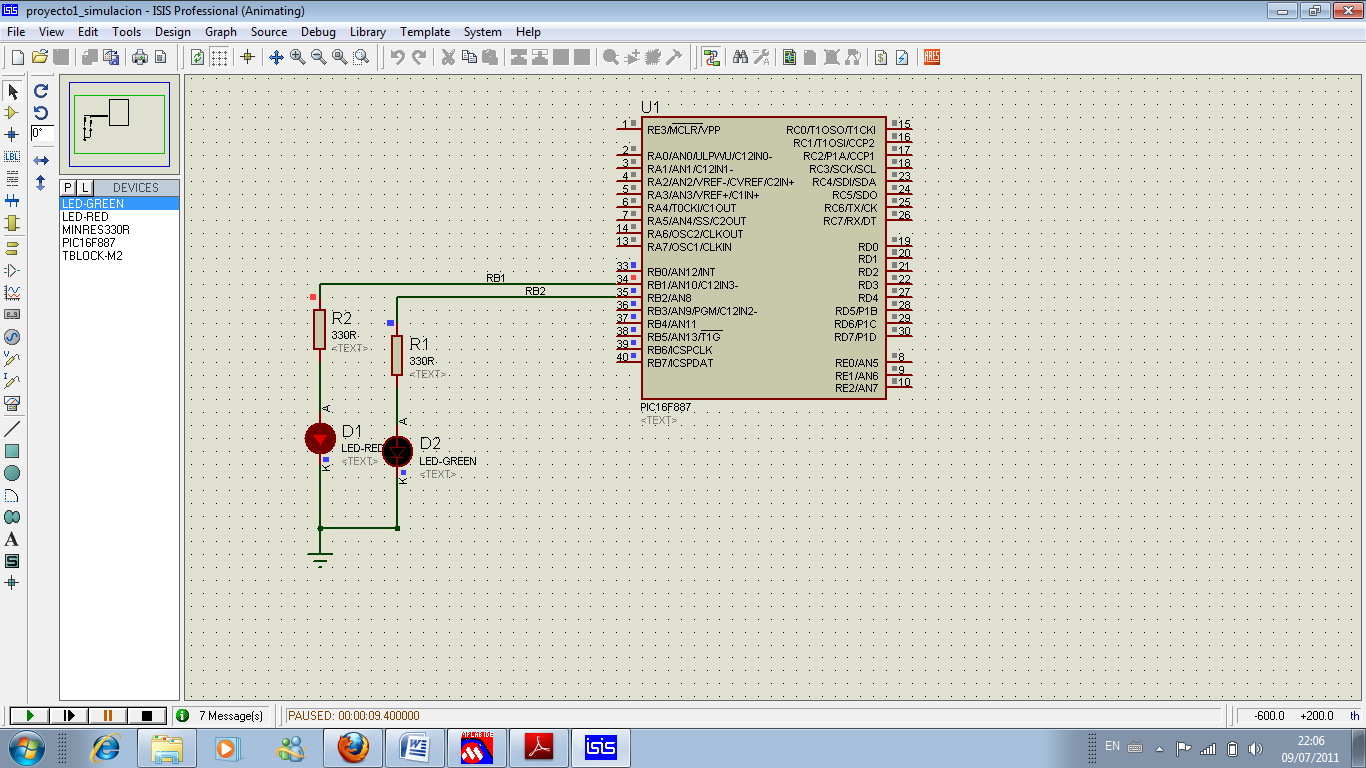
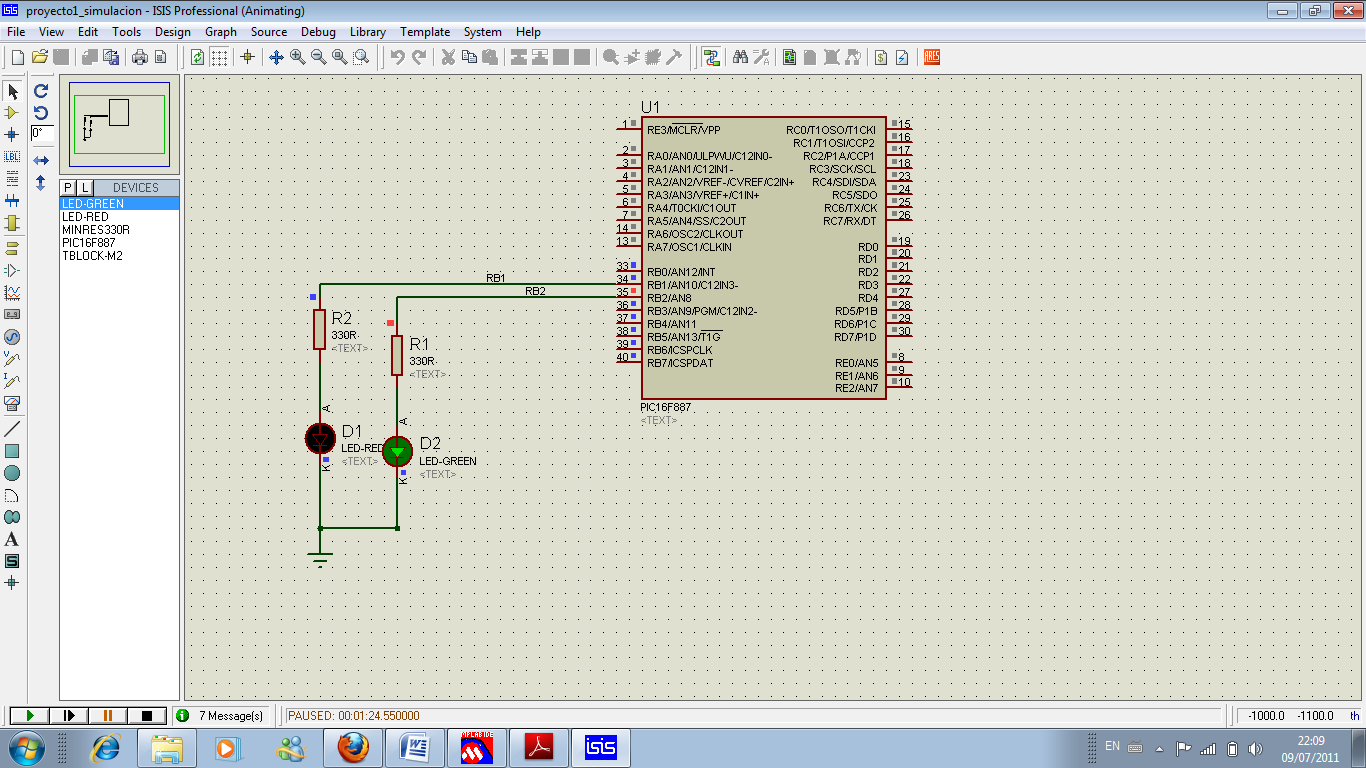
**movlw .197 ;Valor decimal 197**

**movwf TMR0 ;Carga el TMR0 con 217**

**GOTO Loop1**

**END**

1. **COPIA IMPRESA DEL CIRCUITO ARMADO EN PROTEUS**

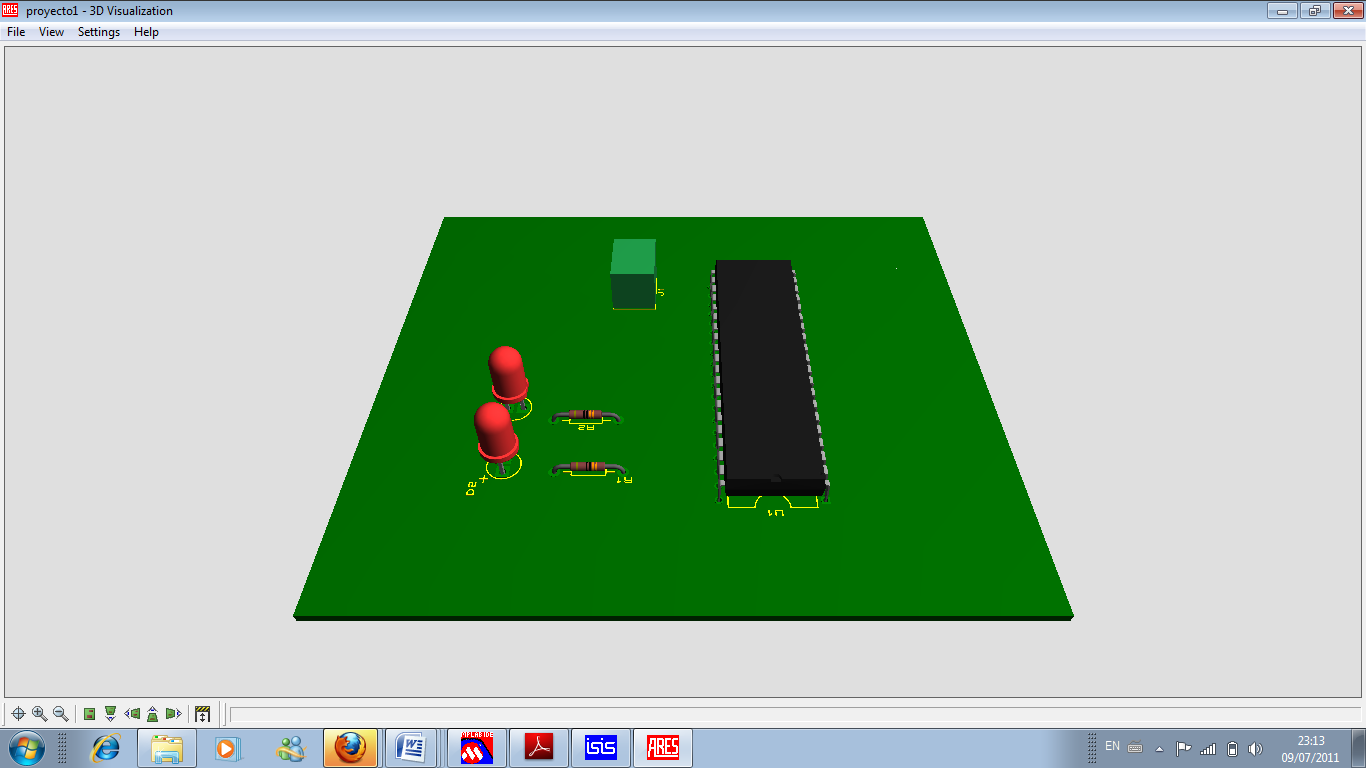
****

1. **CONCLUSIONES:**

* En éste proyecto se volvió a hacer uso del software MPLAB y PROTEUS, lo que ha DEMOSTRADO nuestra capacidad de manejo de estos dos programas, ahora tenemos un desempeño más rápido y eficiente a la hora de usar el MPLAB, para crear un código que nos permita manejar el PIC 16f887 para realizar diferentes tareas en este caso un programa que permite hacer parpadear dos leds con retardos conseguidos con la fórmula del temporizador TMRO.
* Se puede concluir que fue de mucha utilidad las prácticas realizadas en el laboratorio que sirvieron como guía, para la realización del proyecto hablando del caso especifico de un programa que logra retardos con el TMRO para controlar el encendido y apagado de dos leds, además se requirió revisar lo revisado en las clases teóricas en especial lo que es concerniente a los registros y las funciones de cada uno de sus bits, como en este proyecto que se requirió configurar algunos bits del registro STATUS, OPTION e INTCON.
* Es de resaltar detalles que nos encontramos durante la creación de éste proyecto, en lo que concierne a los retardos de tiempo conseguidos con la fórmula del TMRO, cuando se requiere retardos más extensos como en este proyecto que requería retardos de un minuto para cada led, ya que se emplearon contadores de 40,20 y 100 toda instrucción por más que dure 1us con un reloj de 4Mhz debe ser considerada. Pero en este proyecto esas instrucciones le sumaron tiempos en la escala de ms a los retardos de un minuto, que en fin se pueden despreciar.

1. **RECOMENDACIONES:**

* Es recomendable después de tener el código en el MPLAB con su respectivo proyecto, revisar paso a paso el código y para eso es de mucha utilidad la herramienta mencionada anteriormente, la ventana Special Function Registers, que nos muestra el cambio de los valores de los bits a lo largo del programa, y nos muestra en rojo cuando un bit ha cambiado de valor, esto nos ayuda a comprender el comportamiento de los Registros del microcontrolador.
* También se recomienda a la hora de implementar en PROTEUS tener mucho cuidado y sobre todo escoger bien los elementos porque una elección incorrecta representaría una pérdida de tiempo.
* Finalmente se recomiendo el uso del stop watch para hacer el cálculo de los retardos de tiempo y asegurarnos de que nuestro programa funcione. Para la simulación en ISIS no necesitamos la fuente que alimenta el circuito, pues si la utilizamos obtendremos errores.

Anexo