



Universidad Tecnológica del Perú

Facultad de Ingeniería Electrónica y Mecatrónica

Curso : Microcontroladores

Profesor : Domínguez Jesús

Laboratorio Nro : 5

Tema : Timer, Interrupciones y LCD

Integrantes : Espinoza Valera, Jesús Alberto
Francisco
Alberto Francisco, Jaime Félix
Guerrero Isuiza, Mateo

Fecha del Experimento : Lunes, 5 de Noviembre del 2018

Hora : De 9:30 am a 11:00 am

Fecha de entrega del informe : Martes, 6 de Noviembre del 2018

Hora : De 8:00 am a 6:00 pm

2018 - III



Contenido

Introducción.....	3
Capítulo 1: Compendio Teórico	4
1.1. Lenguaje C	4
1.2. MPLAB	4
1.3. PicKit	4
1.4. Copilador XC8	4
1.5. Microcontrolador	4
1.7. LCD	5
Capítulo 2: Resultados Obtenidos	6
2.1. Timer 0 Contador.	6
2.1.1. Diagrama de flujo	6
2.1.2. Código	6
2.1.3. Lenguaje y Simulación.....	7
2.1.4. Circuito Real	8
2.2. Contador Timer 0 con interrupciones.....	9
2.2.1. Diagrama de flujo	9
2.2.2. Código	9
2.2.3. Lenguaje y Simulación.....	10
2.2.4. Circuito Real	11
2.3. Contador Timer0 y LCD	12
2.3.1. Diagrama de flujo	12
2.3.2. Código	12
2.3.3. Lenguaje y Simulación.....	15
2.3.4. Circuito Real	16
Link del Proyecto	17
Observaciones	18
Conclusiones.....	19
Bibliografía.....	20

Introducción

El interés del grupo de investigación se basó a partir de la observación de los distintos programas que se realizaron en las clases de Microcontroladores, donde se puede generar interrupciones en procesos infinitos del PIC18F4550. En consecuencia, nos planteamos la interrogante: ¿En qué medida es posible obtener un temporizador con y sin interrupciones mediante el PIC18F4550? Consideramos que es posible construir un mecanismo que permita al PIC18F4550 ser un circuito temporizador capaz de realizar un ciclo de operaciones y a su vez, pueda añadirse interrupciones para este proceso.

De esta manera, el informe se dividió en cinco partes. En primer lugar se detallan los conceptos de los sistemas o circuitos usados en el proyecto. En segundo lugar se presenta la elaboración del proyecto: El diagrama de flujo, el código de programación en Lenguaje C, las fotos de las simulaciones en Mplab y Proteus, y finalmente las fotos del circuito real programado con el PicKIT 3. En tercer lugar, se muestra el link del video en Youtube. En cuarto lugar, se muestra las observaciones obtenidas en el proyecto. Finalmente, se muestran las conclusiones.

La importancia del proyecto radica en la relevancia de las aplicaciones del PIC para generar saltos en rutinas, es decir, será posible generar programas capaces de reconocer u evitar posibles fallos y situaciones especiales en la programación. Siendo esta proyecto un medio eficaz para evidenciar que el PIC18F4550 puede ser un medio alternativo de circuitos de división de frecuencia. Del mismo modo, el proyecto me permite desarrollar habilidades para la programación y construcción de sistemas eléctricos que serán de gran utilidad en mi carrera profesional como Ingeniero egresado de UTP.

Capítulo 1: Compendio Teórico

1.1. Lenguaje C

El lenguaje C, es un lenguaje que revoluciono ya que debido a su sencillez, y a su tamaño, pero que sobre todo no está dentro de una aplicación en específica, lo hace más potente. C trabaja con tipos de datos que son directamente tratables por el hardware de la mayoría de computadoras actuales, como son los caracteres, números y direcciones. Estos tipos de datos pueden ser manipulados por las operaciones aritméticas que proporcionan las computadoras.

1.2. MPLAB

Este es un programa que sirve para editar distintos micro controladores que este mismo programa puede soportar, sin embargo una de las cosas importantes es que se puede grabar circuitos integrados. Asimismo cabe recalcar que se debe a empezar a escribir el programa, respetando la directiva y/o parámetros necesarios para compilarlo y grabarlo al chip. Definimos directivas como palabras específicas o palabras que han sido separadas para ordenarle al Mplab que funciones debe configurar a la hora de compilar nuestro programa.

1.3. PicKit

El pickit hace posible la programación de micro controlador utilizando todo lo que incluye en el armado del código en el Mplab, desarrollado por la empresa Microchip. Este pickit es muy fácil de conectar ya que solo se conecta mediante una interfaz USB.

1.4. Copilador XC8

Un compilador es un programa informático que permite que el lenguaje avanzado usado por el programador (Lenguaje C, BASIC, etc) sea traducido al lenguaje de máquina, el único que puede leer el microcontrolador. El compilador XC8 es uno de los compiladores de la línea de compiladores MPLAB XC de Microchip, el cual es compatible con los microcontroladores PIC de 8 bits, utilizado en este proyecto.

1.5. Microcontrolador

Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y

ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como microcomputadora. Se puede decir con toda propiedad que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado.

Inicialmente todos los microcontroladores adoptaron la arquitectura clásica de Von Neumann, la cual se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de manera indistinta.

Después, se empezó a realizar microcontroladores con la nueva arquitectura Harvard, la cual se caracteriza por disponer de dos memorias independientes para datos y para instrucciones, permitiendo accesos simultáneos.

Estos microcontroladores necesitan ser programados para poder realizar las funciones. Debido a la gran dificultad de programar los microcontroladores en el código binario, la programación comúnmente se lleva a cabo a través de un lenguaje de alto nivel, es decir, un lenguaje que utilice frases o palabras semejantes o propias del lenguaje humano. Lenguajes como el C, utilizado en nuestro caso, o BASIC son comúnmente utilizados en la programación de los microcontroladores.

1.6. Display 7 segmentos

Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como microcomputadora. Se puede decir con toda propiedad que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado.

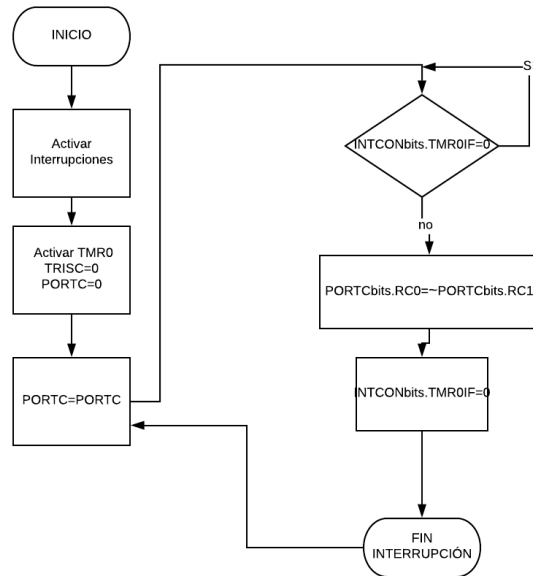
1.7. LCD

Una pantalla de cristal líquido o LCD (sigla del inglés Liquid Crystal Display) es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.

Capítulo 2: Resultados Obtenidos

2.1. Timer 0 Contador.

2.1.1. Diagrama de flujo



2.1.2. Código

/*TMR0 Tiempo 100 mseg con el TMR0, usando 16 bits, Fosc=4Mhz,
Prescaler=8*/

`void TMR0_Init (void)`

```
{
    INTCONbits.GIE=0;
    T0CON=0x80;
    INTCONbits.TMR0IE=1;
    INTCONbits.TMR0IF=0;
}
```

`void main(void)`

```
{
    TMR0_Init();
    TRISC=0;
    PORTC=0;
    TMR0L=15537;
    TMR0H=(15537) >>8;
    while(1)
    {
```

INICIO;

SENSA: if(INTCONbits.TMR0IF==0)goto SENS;

INTCONbits.TMR0IF=0;

PORTCbits.RC0=~PORTCbits.RC0;

goto INICIO;

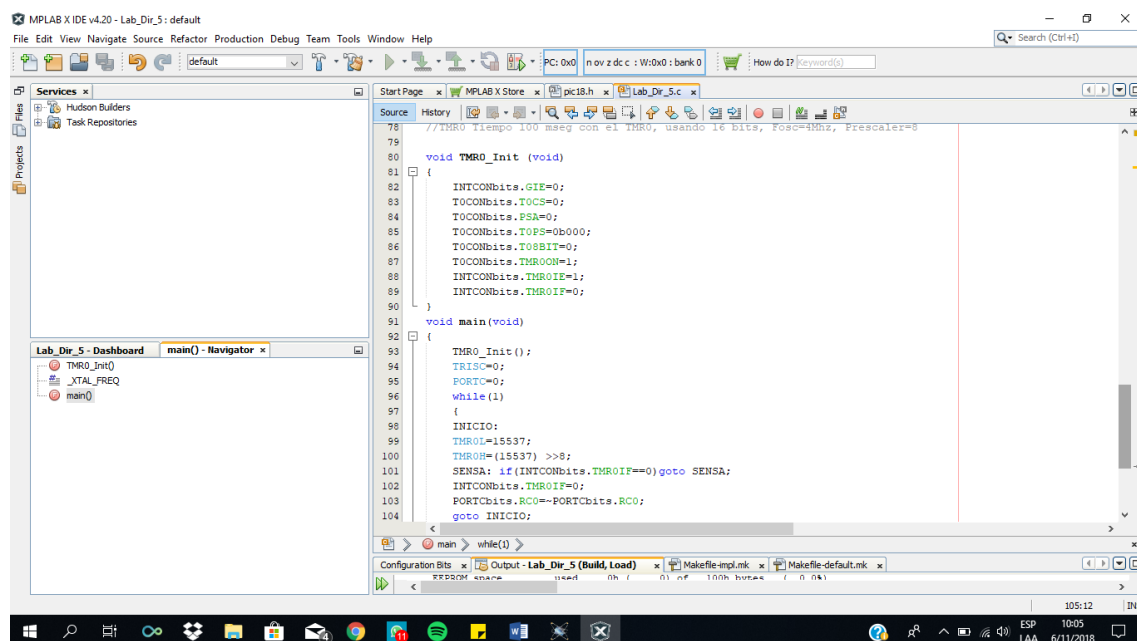
}

return;

}

2.1.3. Lenguaje y Simulación

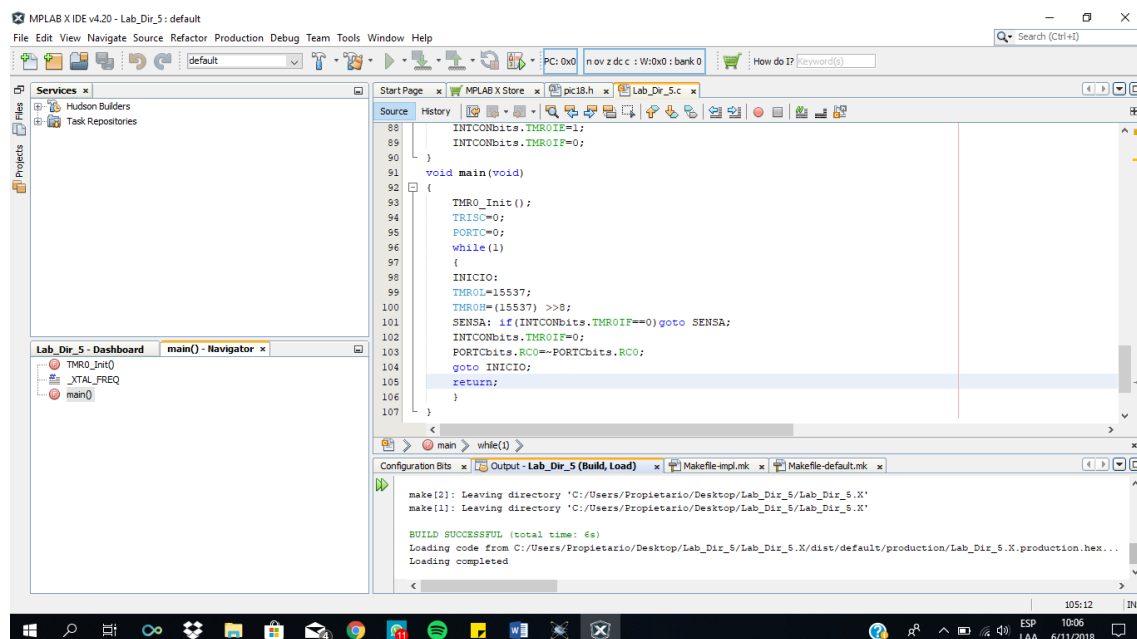
A continuación se muestra el código escrito en MPLAB V4.20:



```

78 //TMR0 tiempo 100 mseg con el TMR0, usando 16 bits, Fosc=4MHz, Prescaler=8
79
80 void TMR0_Init (void)
81 {
82     INTCONbits.GIE=0;
83     T0CONbits.T0CS=0;
84     T0CONbits.PSA=0;
85     T0CONbits.T0PS=0b000;
86     T0CONbits.T08BIT=0;
87     T0CONbits.TMR0ON=1;
88     INTCONbits.TMR0IE=1;
89     INTCONbits.TMR0IF=0;
90 }
91
92 void main(void)
93 {
94     TMR0_Init();
95     TRISC=0;
96     PORTC=0;
97     while(1)
98     {
99         INICIO:
100         TMR0L=15537;
101         TMR0H=(15537) >>8;
102         SENS: if(INTCONbits.TMR0IF==0)goto SENS;
103         INTCONbits.TMR0IF=0;
104         PORTCbits.RC0=~PORTCbits.RC0;
105         goto INICIO;
106     }
107 }

```



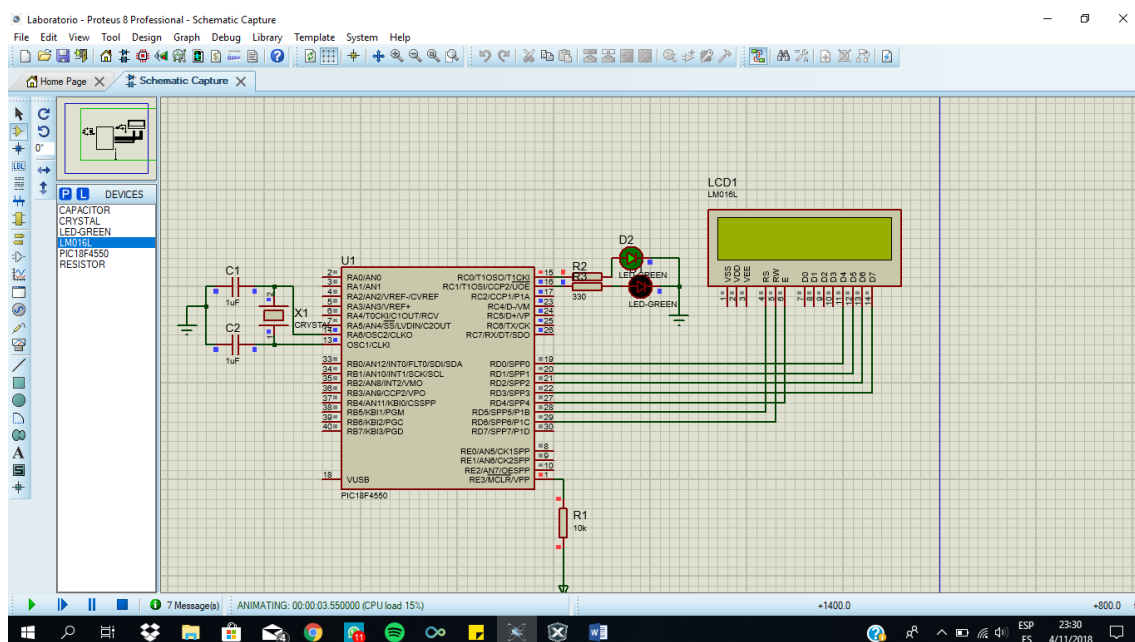
```

make[2]: Leaving directory 'C:/Users/Proprietario/Desktop/Lab_Dir_5/Lab_Dir_5.X'
make[1]: Leaving directory 'C:/Users/Proprietario/Desktop/Lab_Dir_5/Lab_Dir_5.X'

BUILD SUCCESSFUL (total time: 6s)
Loading code from C:/Users/Proprietario/Desktop/Lab_Dir_5/Lab_Dir_5.X/dist/default/production/Lab_Dir_5.X.production.hex...
Loading completed

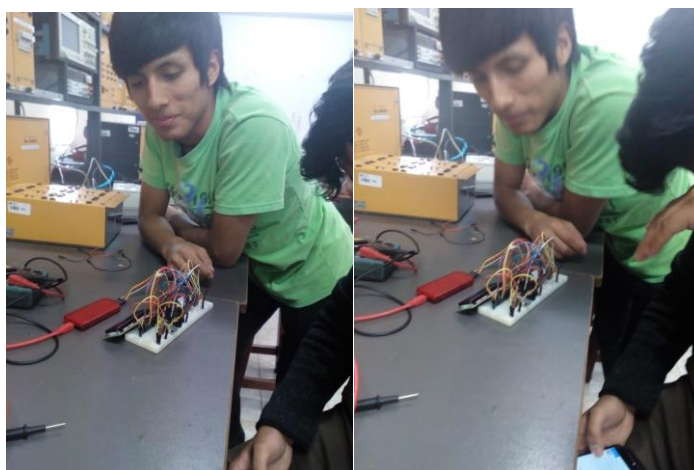
```

Del mismo modo, a continuación se muestra la simulación del circuito construido en Proteus 8:



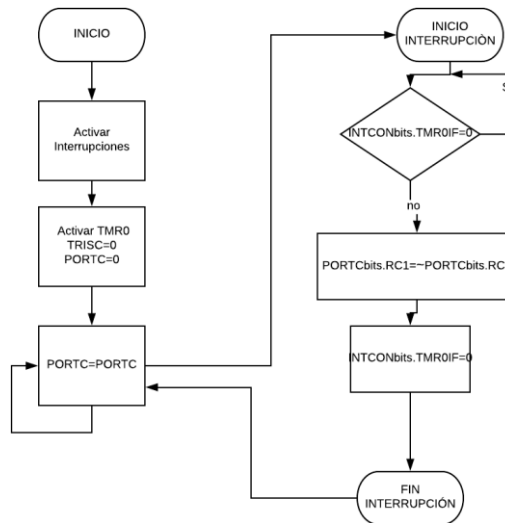
2.1.4. Circuito Real

Para la primera simulación se realizó un circuito en Protoboard conectado a base de la simulación del Proteus y el manual del PIC18F4550. Asimismo, se le colocó un Led, para ver que el circuito funcione correctamente, como se muestra a continuación:



2.2. Contador Timer 0 con interrupciones

2.2.1. Diagrama de flujo



2.2.2. Código

/*TMR0 Tiempo 100 mseg con el TMR0, usando 16 bits, Fosc=4Mhz,
Prescaler=8*/

```
void __interrupt() tc_int (void);
```

```
void TMR0_Init (void);
```

```
void TMR0_Init (void)
```

```
{
    INTCONbits.GIE=0;
    T0CON=0x80;
    INTCONbits.TMR0IE=1;
    INTCONbits.TMR0IF=0;
    INTCONbits.PEIE=1;
    INTCONbits.GIE=1;
}
```

```
void main(void)
```

```
{
    TMR0_Init();
    TRISC=0;
    PORTC=0;
```

```

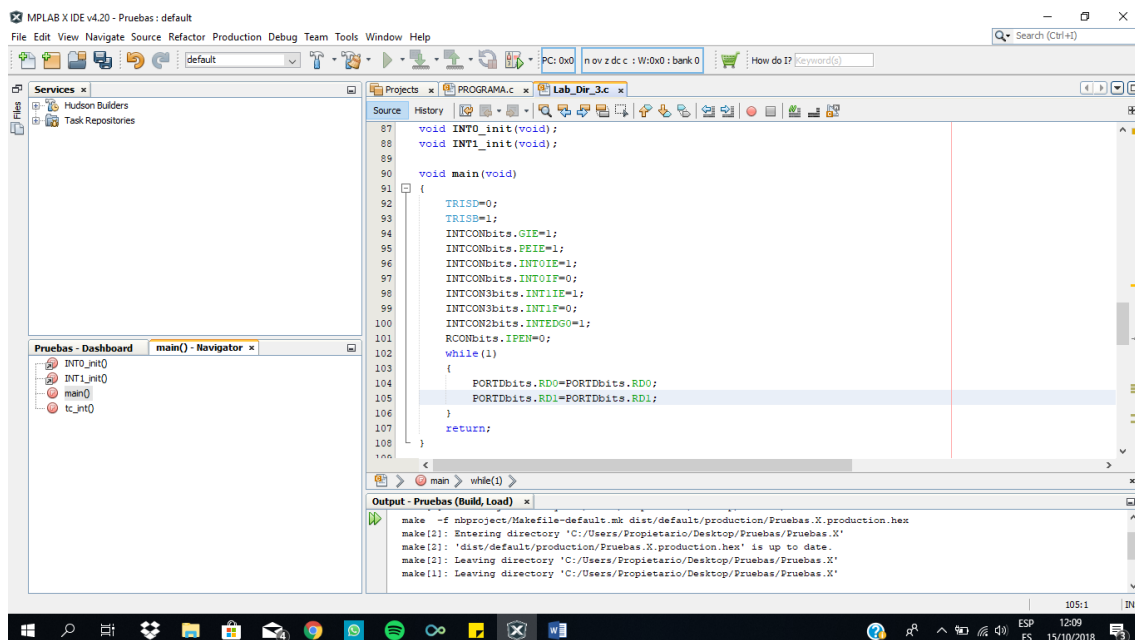
TMR0L=15537;
TMR0H=(15537) >>8;
while(1)
{
}
return;
}

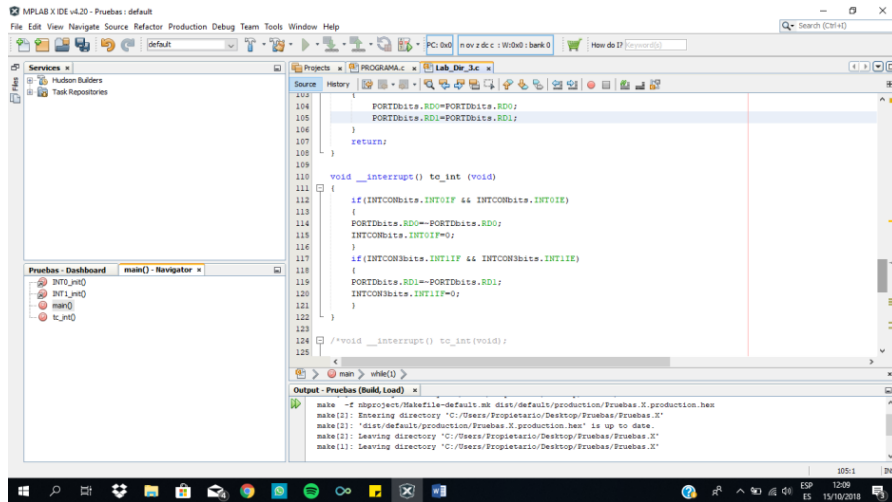
void __interrupt() tc_int (void)
{
    if (INTCONbits.TMR0IF==1)
    {
        PORTCbits.RC1=~PORTCbits.RC1;
        INTCONbits.TMR0IF=0;
    }
}

```

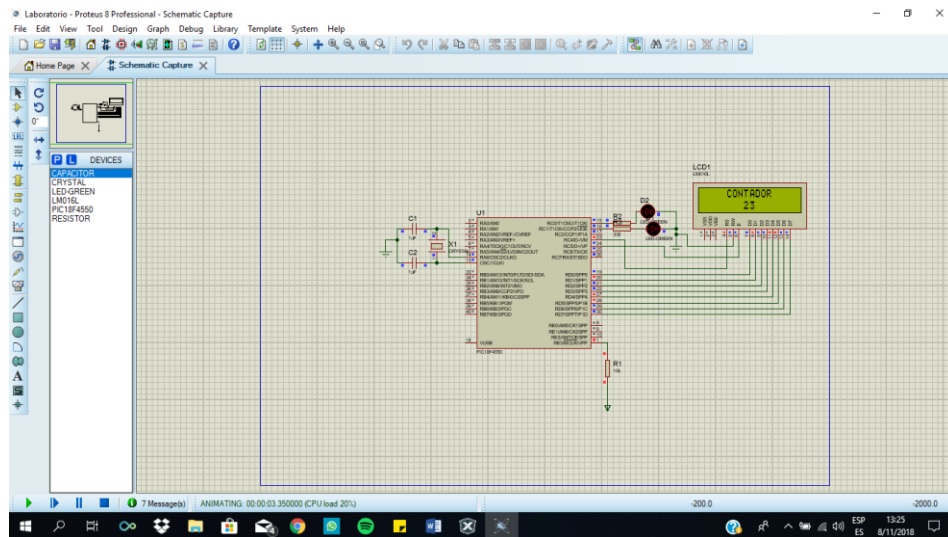
2.2.3. Lenguaje y Simulación

A continuación se muestra el código escrito en MPLAB V4.20:



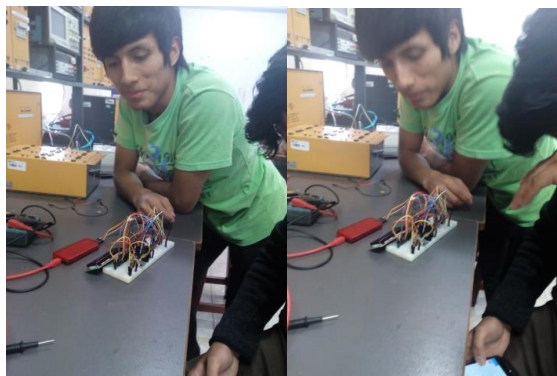


Del mismo modo, a continuación se muestra la simulación del circuito construido en Proteus 8:



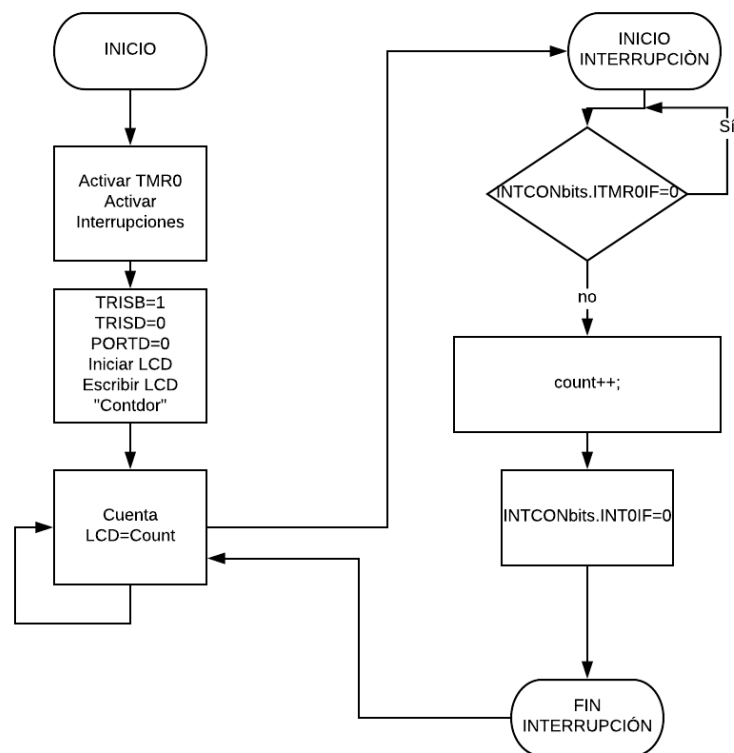
2.2.4. Circuito Real

Para la segunda simulación se realizó un circuito en Protoboard conectado a base de la simulación del Proteus y el manual del PIC18F4550. Asimismo, se le colocó un Led, para ver que el circuito funcione correctamente, como se muestra a continuación:



2.3. Contador Timer0 y LCD

2.3.1. Diagrama de flujo



2.3.2. Código

/*TMR0 Tiempo 100 mseg con el TMR0, usando 16 bits, Fosc=4Mhz,
Prescaler=8*/

```
void __interrupt() tc_int (void);
```

```
void TMR0_Init (void);
```

```
int count;
```

```
void TMR0_Init (void)
```

```
{
```

```
    INTCONbits.GIE=0;
```

```
    TOCON=0x80;
```

```
    INTCONbits.TMR0IE=1;
```

```
    INTCONbits.TMR0IF=0;
```

```
    INTCONbits.PEIE=1;
```

```
    INTCONbits.GIE=1;
```

```
}
```



```
void main(void)
{
    TMR0_Init();
    int c[4]={0x01,0x38,0x0C,0x06};
    int m[12]={' ',' ',' ',' ','C','O','N','T','A','D','O','R'};
    int a,d,u;
    TRISC=0;
    TRISD=0;
    PORTC=0;

    PORTCbits.RC4=0;
    PORTCbits.RC5=0;
    __delay_ms(10);

    for(a=0;a<4;a++)
    {
        PORTD=c[a];
        __delay_ms(10);
        PORTCbits.RC5=1;
        __delay_ms(10);
        PORTCbits.RC5=0;
        __delay_ms(10);
    }

    PORTCbits.RC4=1;
    PORTCbits.RC5=0;
    __delay_ms(10);

    for(a=0;a<12;a++)
    {
        PORTD=m[a];
        __delay_ms(10);
        PORTCbits.RC5=1;
        __delay_ms(10);
```



```
    PORTCbits.RC5=0;  
    __delay_ms(10);  
}
```

```
TMR0L=3036;  
TMR0H=(3036) >>8;  
while(1)  
{  
    d=count/10;  
    u=count%10;  
    PORTCbits.RC4=0;  
    PORTCbits.RC5=0;  
    __delay_ms(10);  
  
    PORTD=0xC7;  
    __delay_ms(10);  
    PORTCbits.RC5=1;  
    __delay_ms(10);  
    PORTCbits.RC5=0;  
    __delay_ms(10);  
  
    PORTCbits.RC4=1;  
    PORTCbits.RC5=0;  
    __delay_ms(10);  
  
    PORTD=d+0x30;  
    __delay_ms(10);  
    PORTCbits.RC5=1;  
    __delay_ms(10);  
    PORTCbits.RC5=0;  
    __delay_ms(10);  
  
    PORTD=u+0x30;  
    __delay_ms(10);
```

```

PORTCbits.RC5=1;

__delay_ms(10);

PORTCbits.RC5=0;

__delay_ms(10);

}

return;

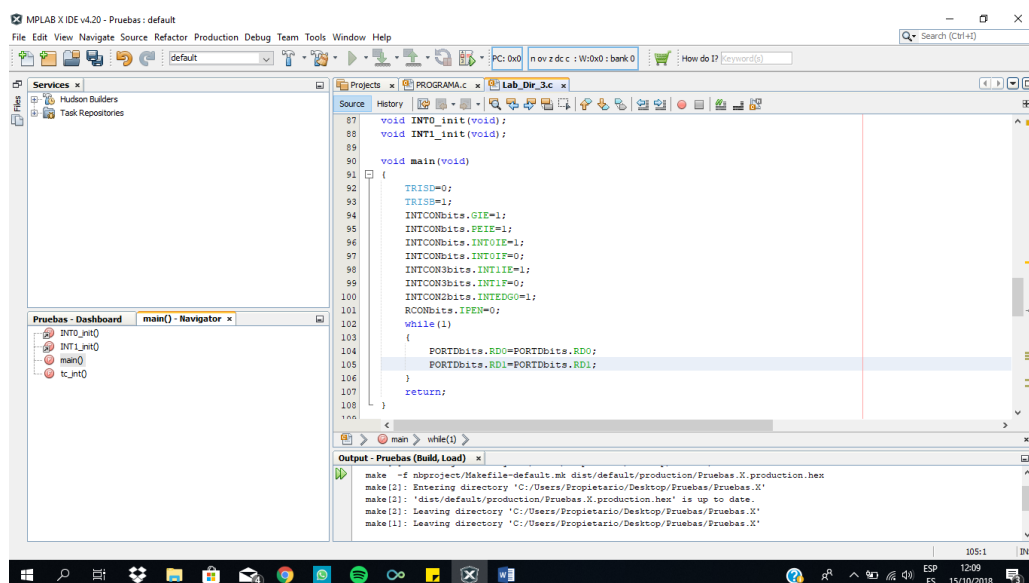
}

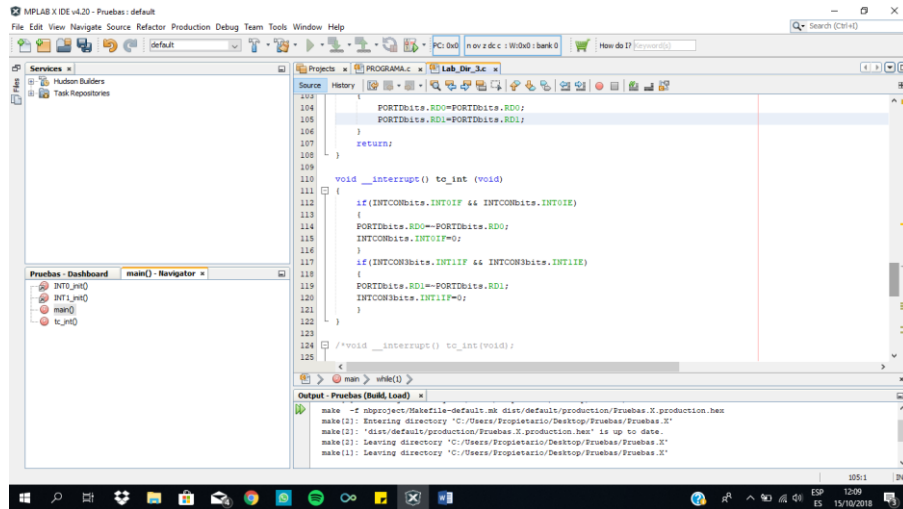
void __interrupt() tc_int (void)
{
    if (INTCONbits.TMR0IF==1)
    {
        count++;
        if(count==100)
        {
            count=0;
        }
        PORTCbits.RC1=~PORTCbits.RC1;
        INTCONbits.TMR0IF=0;
    }
}

```

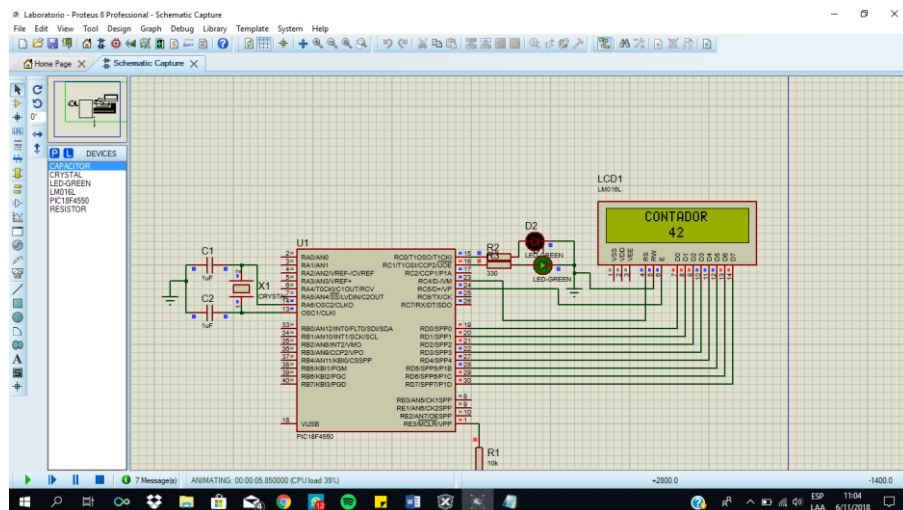
2.3.3. Lenguaje y Simulación

A continuación se muestra el código escrito en MPLAB V4.20:



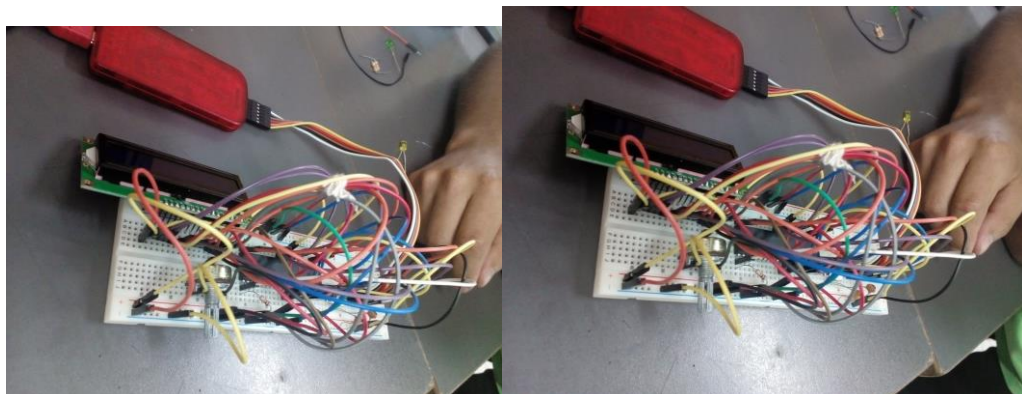


Del mismo modo, a continuación se muestra la simulación del circuito construido en Proteus 8:



2.3.4. Circuito Real

Para la segunda simulación se realizó un circuito en Protoboard conectado a base de la simulación del Proteus y el manual del PIC18F4550. Asimismo, se le colocó un Led, para ver que el circuito funcione correctamente, como se muestra a continuación:





Link del Proyecto

[https://www
.youtube.co
m/watch?v=
ja5Yq_6kAv
k&feature=y
outu.be](https://www.youtube.com/watch?v=ja5Yq_6kAvk&feature=youtu.be)

Observaciones

- Fue fundamental activarse los 16 bits del TMR0, se tendría que hacer un bucle en el cual, las interrupciones reconocerán la primera vez que el INT0IF.TMR0 cambia a 1 y se cortará el bucle para el tiempo. Es decir, si yo tengo un ciclo de 100 repeticiones para 1 mseg, se tendrá 100 veces el cambio de valor del INT0IF.TMR0 a 1, y al momento de activar la interrupción, esta se activará con la primera vez que INT0IF.TMR0 tome 1, por lo que se tendrá un tiempo de 1mseg y no uno de 100 mseg.
- Si se deja un bucle infinito dentro del while(1), entonces las interrupciones consumirán mayor CPU del Pic, por lo que el proceso no será el óptimo y se tendrá un tiempo de retraso, por lo que para la Actividad 2 y 3 se tuvo que eliminar el bucle del RC0 del While(1).

Conclusiones

- La configuración de los bits del PIC18F4550, en MPLABX, es esencial en la simulación ya que de hacerlo de manera correcta nos la facilitara, pero haciéndolo mal generará un error.
- Es fundamental escribir cada detalle del lenguaje C del programa de manera precisa para evitar errores.
- El regreso del bit INT0IF.TMR0 a cero, durante la interrupción es muy esencial ya que de no hacerlo la interrupción no entenderá que se ha realizado y no se podrá volver un ciclo infinito.
- Para poder usar la interrupción se tuvo que activar los 16 bits del Timer 0.
- Es necesario determinar el valor de TMR0L para poder obtener un temporizador con el tiempo base que se desee, en este caso fue de 1 mseg.



Bibliografía

- MICROCHIP (2018) (<https://www.microchip.com/>) Sitio web oficial de Microchip; contiene información acerca del compilador XC8 y de MPLABX.
- ELECTRONICA ESTUDIO (2018)
(<http://www.electronicaestudio.com/index.htm>) Sitio web oficial de Electrónica estudio; contiene información acerca de los microcontroladores.
- WIKIPEDIA (2018)(<https://es.wikipedia.org/wiki/Wikipedia:Portada>) Sitio web oficial de Wikipedia; contiene información de interés.