INFORME III – CONTADOR CONVERSOR

Fabián Estiven Zea González Cód.: 201210733, Jhonatan Reinaldo Gómez Pesca Cód.: 201210146

Universidad Pedagógica y Tecnológica de Colombia

14 de Marzo de 2016

Fabian.zea@uptc.edu.co

jhonatan.gomez@uptc.edu.co

***Resumen- En este laboratorio se realizara un contador basado en lenguaje ensamblador (assembler), el cual ira contado de cero hasta 255 en BCD; una vez el conteo llegue hasta el final, se debe iniciar una secuencia de luces que debe durar mínimo un minuto. El contador debe contar una opción de carga de datos y una opción de reinicio, por lo cual si el contador termina la secuencia de luces no debe comenzar desde cero, si no desde el valor que fue cargado anteriormente, si no se cargó ningún valor, el conteo al finalizar la secuencia debe comenzar desde cero. Todo esto se programara en lenguaje ensamblador sobre el pic16f887***

***Índice de términos: assembler, pic16f887, MPLABX***

# INTRODUCCIÓN

El lenguaje ensamblador, o assembler (en inglés assembly lenguaje y la abreviación asm), es un lenguaje de programación de bajo nivel. Consiste en un conjunto de mnemónicos que representan instrucciones básicas para los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura de procesador y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador. Cada arquitectura de procesador tiene su propio lenguaje ensamblador que usualmente es definida por el fabricante de hardware, y está basada en los mnemónicos que simbolizan los pasos de procesamiento (las instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje.

Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual). Esto está en contraste con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portátiles. Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina. Muchos sofisticados ensambladores ofrecen mecanismos adicionales para facilitar el desarrollo del programa, controlar el proceso de ensamblaje, y la ayuda de depuración. Particularmente, la mayoría de los ensambladores modernos incluyen una facilidad de macro (descrita más abajo), y se llaman macro ensambladores.

Fue usado principalmente en los inicios del desarrollo de software, cuando aún no se contaba con potentes lenguajes de alto nivel y los recursos eran limitados. Actualmente se utiliza con frecuencia en ambientes académicos y de investigación, especialmente cuando se requiere la manipulación directa de hardware, alto rendimiento, o un uso de recursos controlado y reducido. También es utilizado en el desarrollo de controladores de dispositivo (en inglés, device drivers) y en el desarrollo de sistemas operativos, debido a la necesidad del acceso directo a las instrucciones de la máquina. Muchos dispositivos programables (como los microcontroladores) aún cuentan con el ensamblador como la única manera de ser manipulados.

# OBJETIVOS

* Realizar un contador binario en BCD
* Generar una secuencia de luces de mínimo un minuto de duración
* Descargar el programa en el pic16f887 para ver su funcionamiento físico.

# PROCEDIMIENTO

Para el desarrollo de este contador en lenguaje ensamblador es necesario primero crear un proyecto en MPLABX, donde MPLAB es el entorno de programación de los microcontroladores de MICROCHIP, para este caso se usara el pic16f887, por esto es muy importante al momento de crear el proyecto, definir muy bien el tipo de microcontrolador a utilizar; dado que para cada microcontrolador existen diferentes configuraciones que son muy importantes realizar al momento de comenzar con un nuevo proyecto.

El datasheet del pic16f887 brindara una importante ayuda al momento de desarrollar la guía, dado que en él se encuentran todas las palabras de configuración, configuración de puertos y las diferentes listas de instrucción.

Para este primer laboratorio se comienza importando cada una de las diferentes librerías que se van a implementar, esto con el fin de tener un mejor conocimiento sobre la manera en que funciona el microcontrolador, así solo por esta ocasión no se usara la instrucción: #include "p16f887.inc". Por lo tanto las librerías usadas se muestran en la siguiente figura:

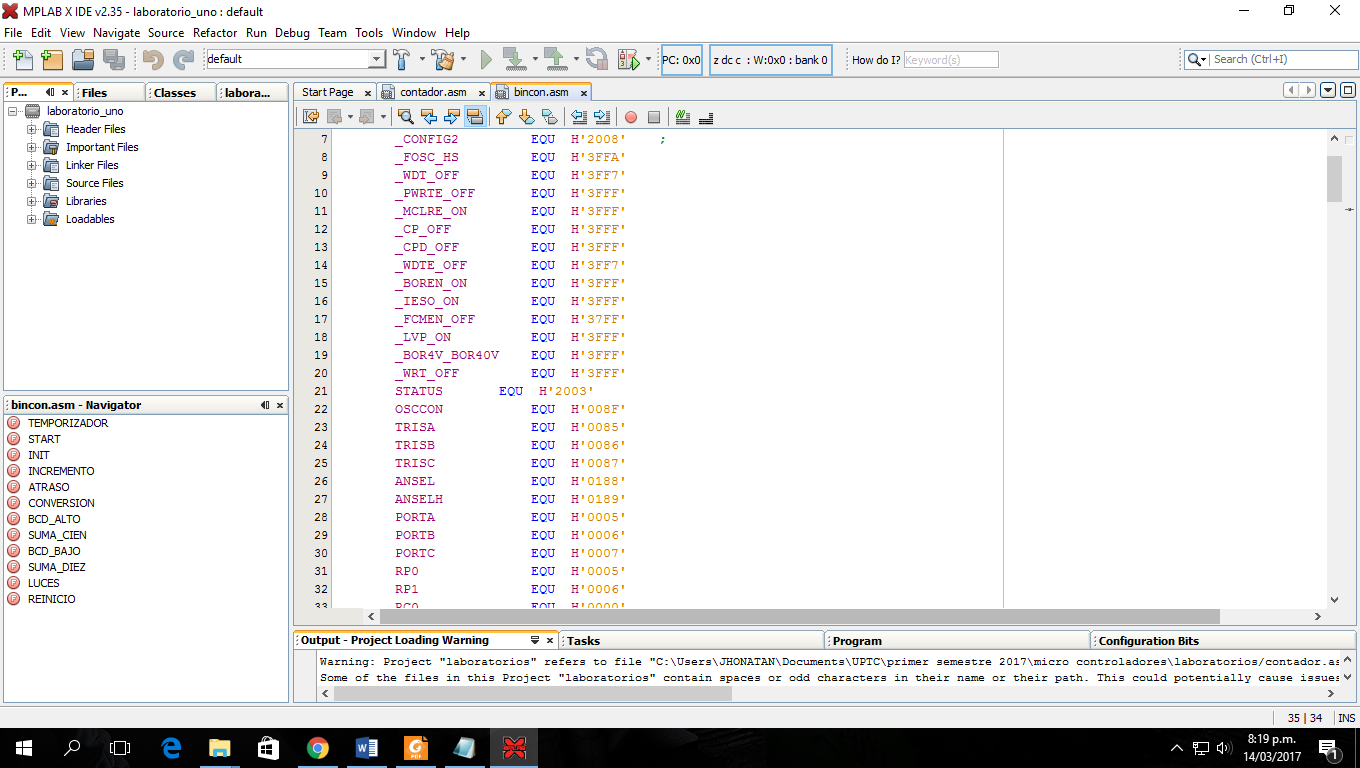


Fig. 1 librerías pic16f887

Una vez llamadas las respectivas librerías que se requieren para el correcto funcionamiento del microcontrolador, se debe entonces posteriormente escribir las palabras de configuración respectivas. Como se muestra en la figura 2.

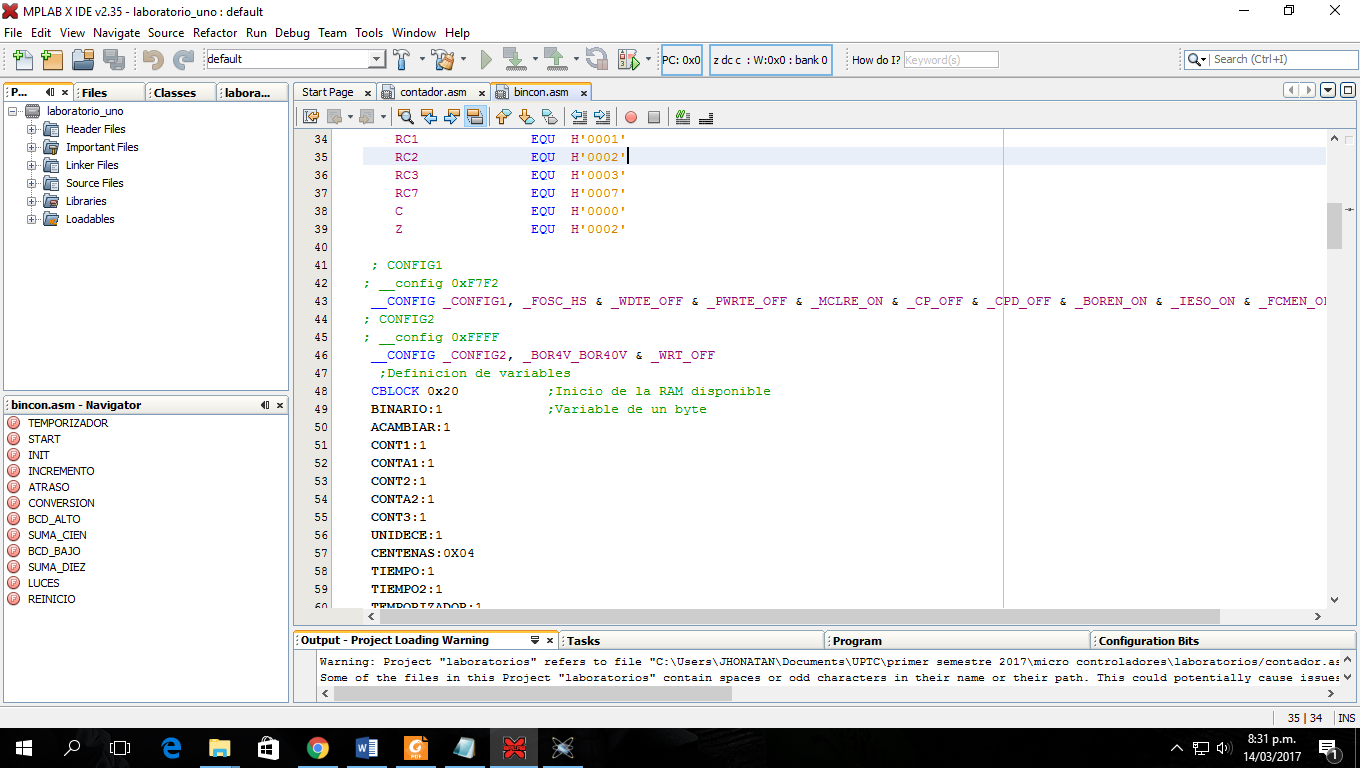


Fig. 2 palabras de configuración del pic.

Las palabras de configuración me definen parámetros tales como, tipo de oscilador, permisos de escritura y lectura entre otros parámetros.

Una vez definidas las palabras de configuración se procede con la configuración de los puertos. El pic16f887 cuenta con 4 puertos (puerto A, B, C y D) los cuales pueden ser definidos como entradas análogas o digitales, o si se desea se puede configurar solamente como una salida digital; para este caso se usara el puerto A como entrada digital, será usado para cargar un dato binario de 8 bits, la parte baja del puerto B será usado como salida digital donde los 4 bits más bajos representaran las unidades, la parte alta del puerto B representara las decenas y finalmente la parte baja del puerto C representara las centenas; la parte restante del puerto C se usara para conectar el respectivo pulsador para cargar los datos. La forma de configurar los puertos se muestra en la figura 3.

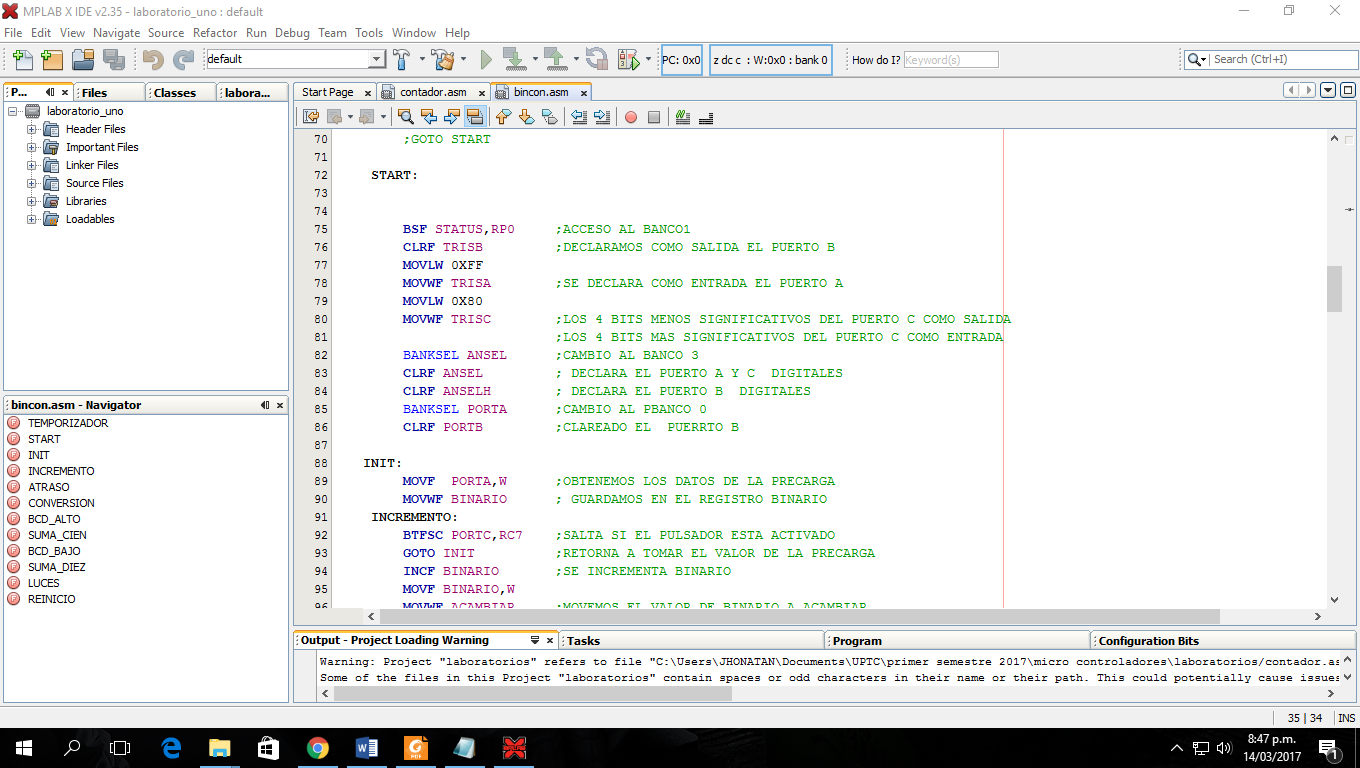


Fig. 3 configuración de los puertos.

La configuración de los puertos del microcontrolador consta básicamente en acceder al banco donde se encuentran los respectivos triestados de cada puerto, el triestado de un puerto cumple la función de definirlo como entrada o como salida. Si el triestado todos los bits se encuentran en 1, el puerto queda definido como entrada, pero sí en cambio los bits del triestado se encuentran en cero, el puerto queda entonces definido como salida, ahora para definir si el puerto al ser definido como entrada funcionara como entrada digital o si en cambio funcionara como entrada análoga; para el caso en el que el puerto sea definido como salida, este puerto solo podrá ser de salida digital dado que el pic16f9887 solo posee salida digital.

Para la configuración del puerto al ser definido como entrada, se debe configurar entonces el registro ANSEL, que se encarga de definir el puerto como entrada análoga o digital, si el registro ANSEL se encuentra en cero, todos los pines del puerto serán definidos de modo que funcionaran solamente de forma digital, caso contrario ocurre si ANSEL se inicia con unos, en este caso el puerto funcionara como entrada análoga.

Cada puerto posee 8 pines, los cuales pueden ser configurados de forma independiente, unos pueden ser configurados como entradas digitales otros como entradas análogas y otros como salidas digitales, y esto se logra configurando bit a bit cada uno de los pines del puerto.

Para la creación de variables se debe tener en cuenta que existen dos tipos de registros en el microcontrolador, y estos son los registros de propósito general y los registros de propósito específico; los registros de propósito específico no pueden ser utilizados como variables, dado que estos son usados para la configuración interna del microcontrolador por tal razón no es recomendable usarlos para otros propósitos, por tal razón el integrado cuenta con una gran cantidad de registros de propósito general que permiten usarlos de la manera en la que el usuario requiera.

Por tal razón las variables que se empleen para el desarrollo de esta guía comenzaran desde la posición 0X20 que es la posición desde la cual comienzan los registros de propósito específico según el datasheet.

Además de configurar los puertos, las palabras de configuración y la posición de los registros de propósito específico es de vital importancia configurar el oscilador del microcontrolador. Este puede ser interno o externo al microcontrolador, pero es requerido para el correcto funcionamiento del microcontrolador, dado que si el oscilador no es configurado este no realizara ninguna acción dado que el integrado funciona de igual manera que un computador por lo que necesita de una señal de reloj para realizar cada una de las operaciones. En la figura 4 se ve la forma en la que fue configurado el oscilador del microcontrolador.

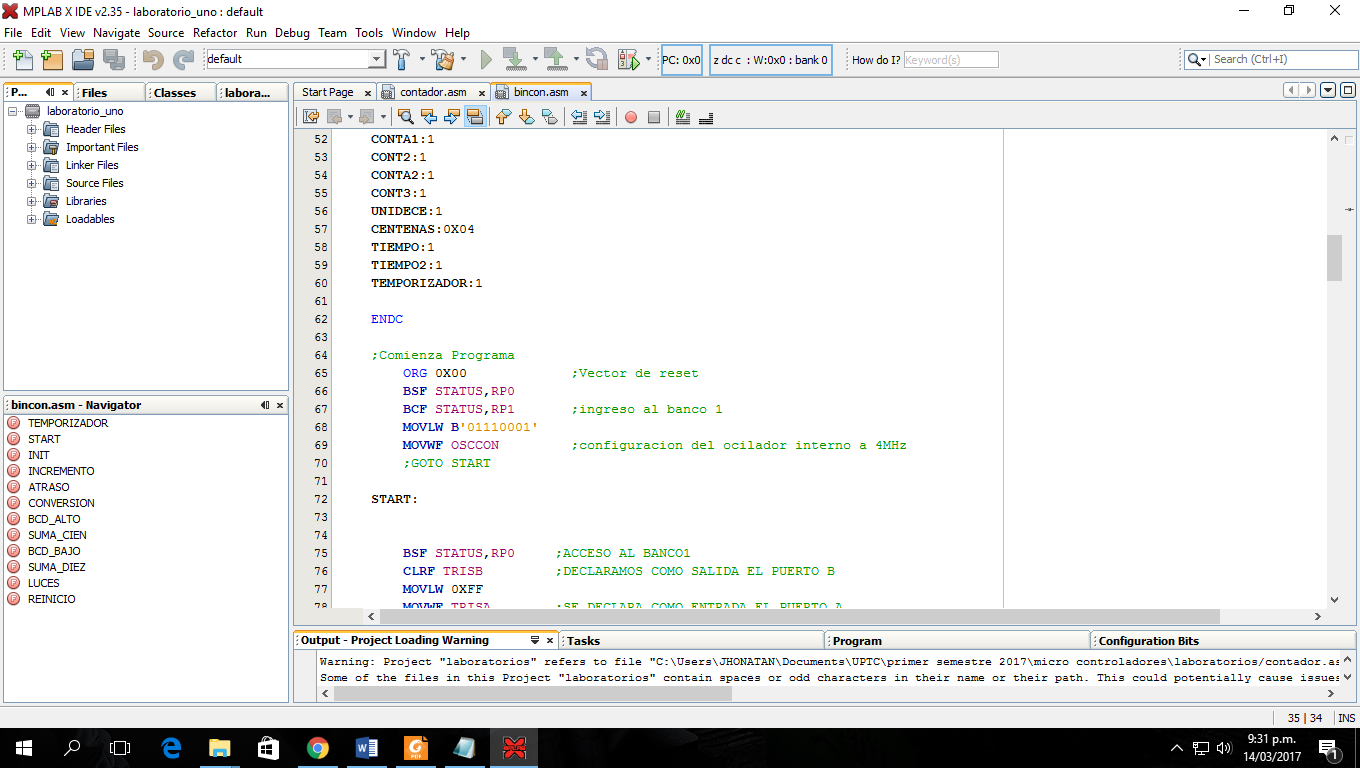


Fig. 4 configuraciones del oscilador.

El registro que se emplea para la configuración es OSCCON, este registro se encarga de controlar un multiplexor, el cual se puede ver en la figura 5. Los bits del 4 al 6 se encargan de controlar este multiplexor, el cual me elige entre 8 posibles valores de oscilador, aparte de eso con el registro OSCCON se puede escoger entre el oscilador interno que viene con el integrado, o por el contrario se puede elegir usar un oscilador externo.

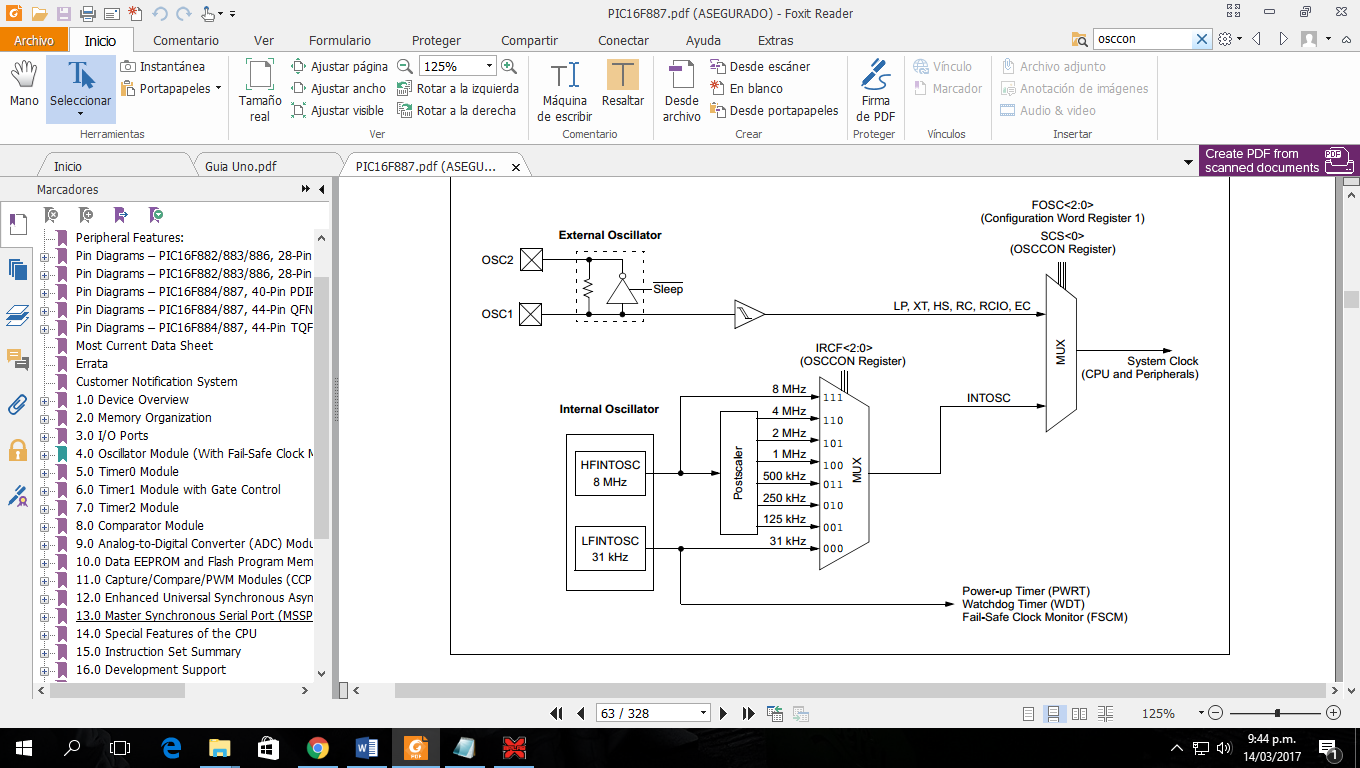


Fig. 5 configuraciones del oscilador.

Como se puede ver en la figura 5 se muestran todas las posibles frecuencias que se pueden utilizar como frecuencia de oscilación. Además también se puede ver como se conecta un oscilador externo.

Una vez ya configurado el microcontrolador de la manera deseada se procede a realizar el contador binario, cabe aclarar que este contador comenzara contando en binario y seguidamente el contador pasara a una subrutina que se encargara de la conversión del dato, pasándolo de binario a BCD. El código implementado para el contador se muestra en la siguiente figura.

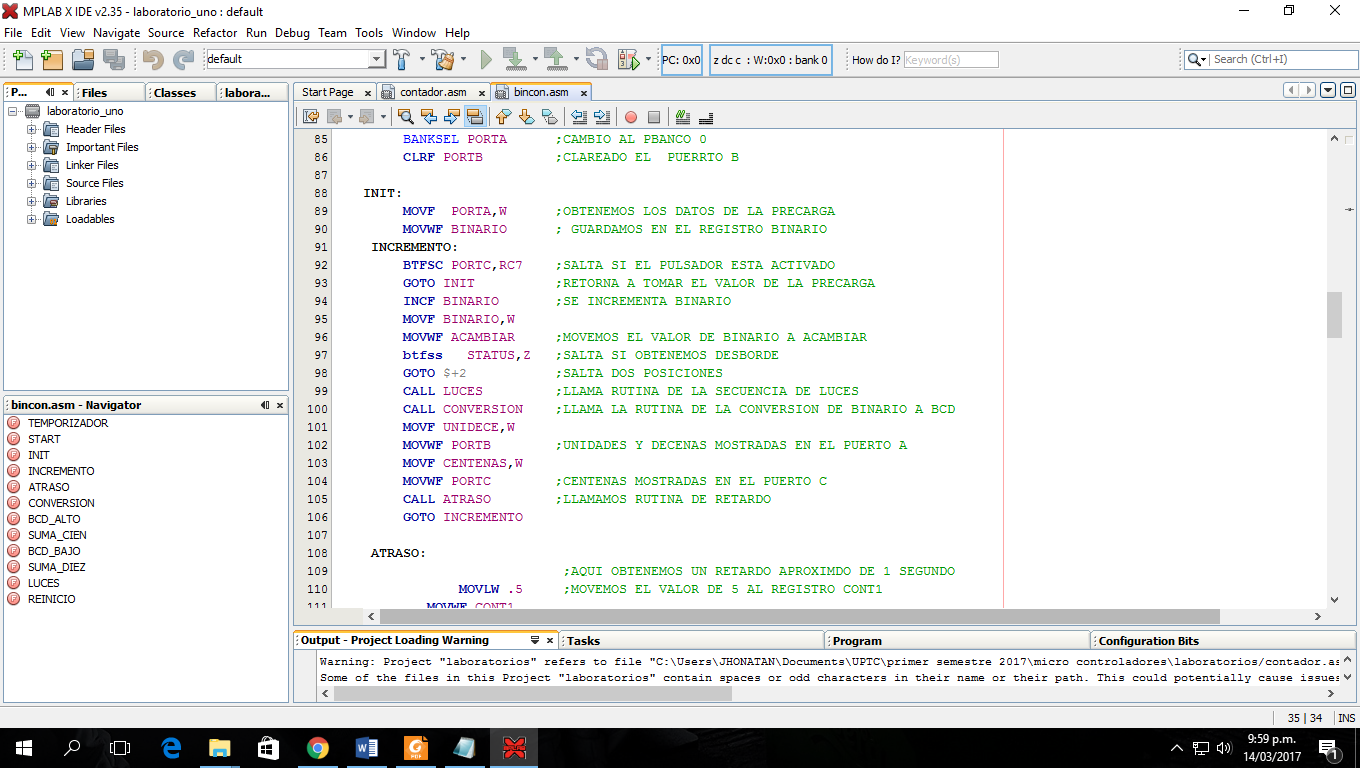


Fig. 6 contador binario.

En la figura 6 se puede apreciar los comandos necesarios para la realización del contador, en este caso también se incluyen el llamado de otras subrutinas que se encargan de aspectos como el retardo (que se tratara más adelante) y del llamado de la subrutina de conversión.

El contador simplemente se basa en la instrucción INCF la cual se encarga de aumentar el valor de un registro, y al escribirlo de la manera INCF binario lo que se hace es incrementar el valor binario y guardarlo en el mismo registro en ese caso es como hacer un cont=cont+1 en el caso del lenguaje c.

Una vez hecho el incremento en la variable binario, el dato se cambia al registro de trabajo W donde después de envía a la subrutina de conversión la cual se encarga de convertir el numero binario que le llega a su equivalente en BCD.

La subrutina de conversión llamada CONVERSION usa un método de conversión se hace realizando divisiones mediante restas sucesivas. Primero entre 100 para obtener el nibble bajo de BCDH y después entre 10 para obtener el nibble alto de BCDL y el residuo pasa a formar el nibble bajo de BCDL. El código de conversión es el mostrado en la figura 7.

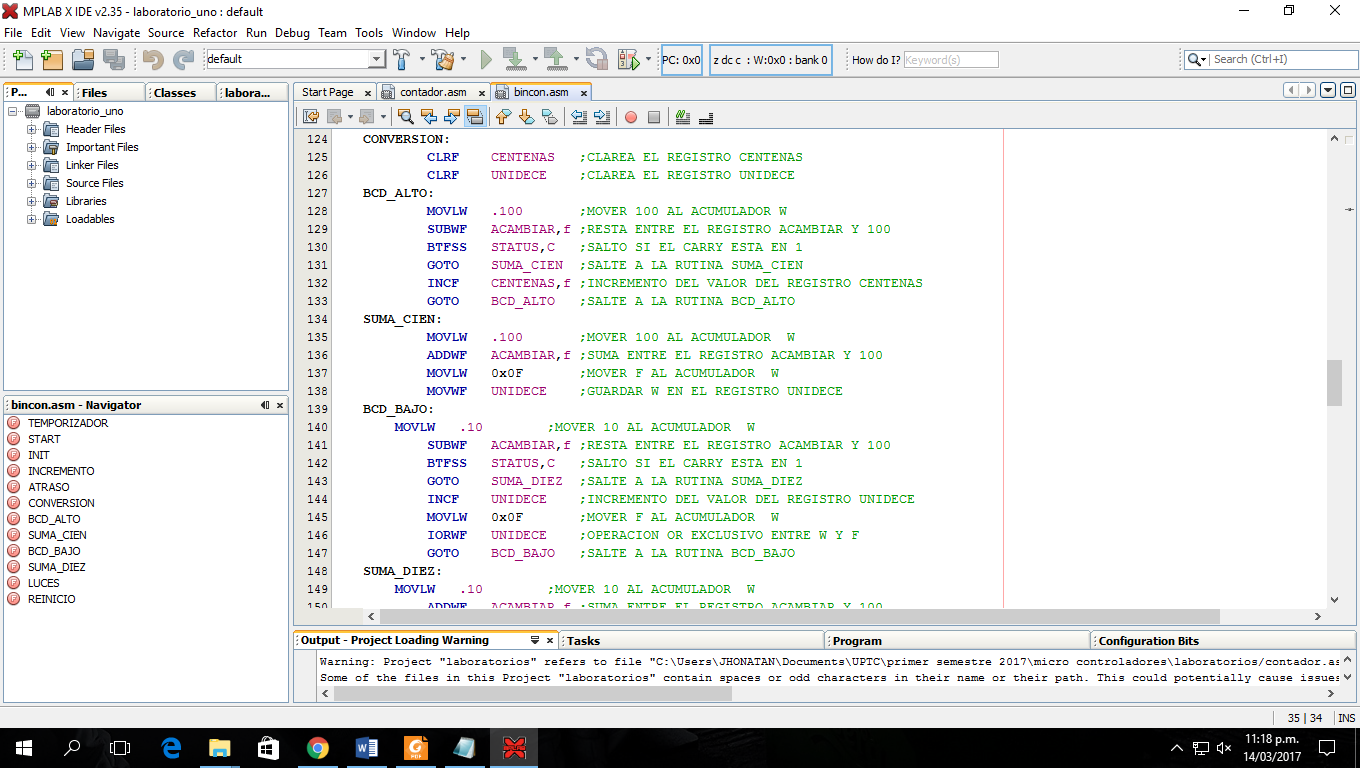


Fig. 7 subrutina de conversión

Como la frecuencia de trabajo del microcontrolador es muy alta, los cálculos serán muy rápidos y por ende los cambios en la salida de los puertos serán muy rápidas y es por eso que se emplean retardos por software que son una especia de bucle que se toma determinado tiempo en ejecutar y que cumple la función de ralentizar el micro en ese bucle por un determinado periodo de tiempo. El retardo por software se puede ver en la figura 8.

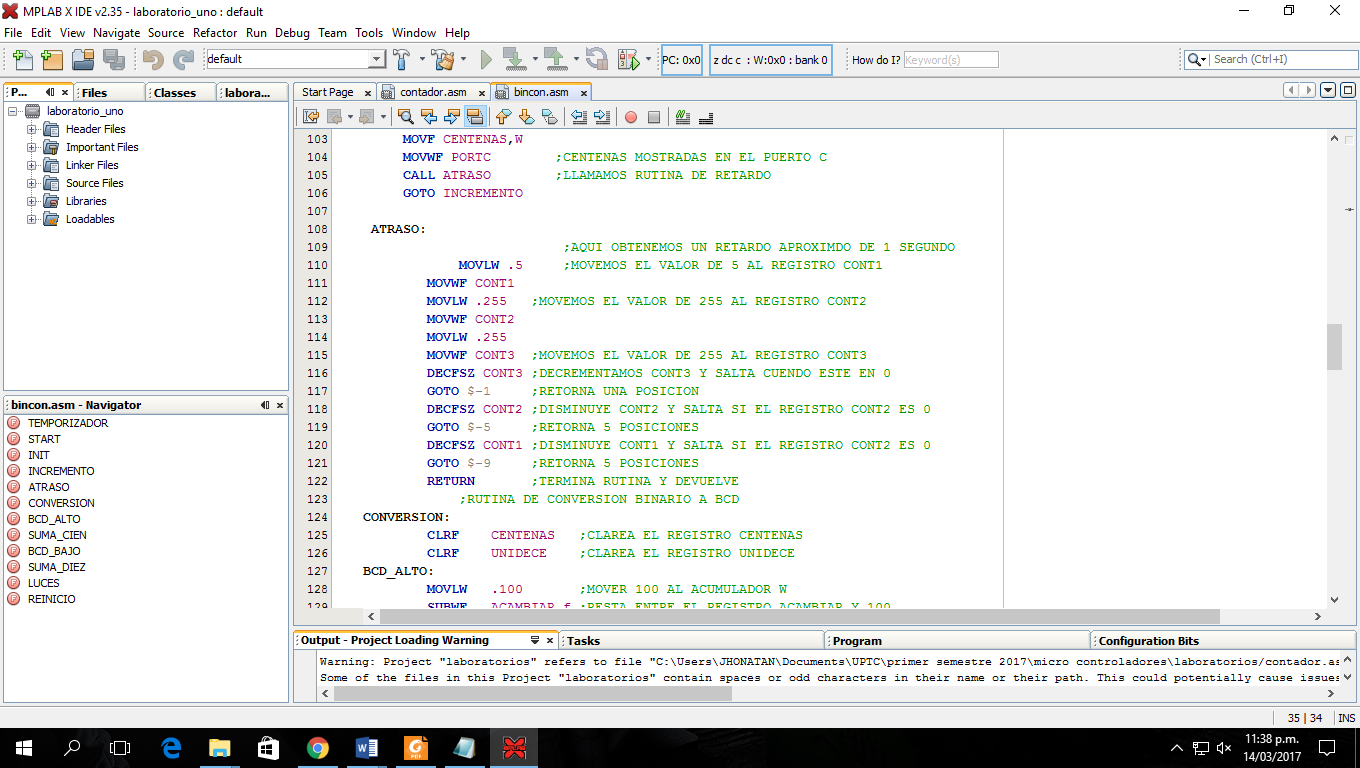


Fig. 8. Retardo por software.

La manera para calcular los retardos se hace de la siguiente manera:

Retardo para el incremento del número en el contador

La frecuencia del oscilador es de 4MHz

CONT1=5

CONT2=255

CONT3=255

Retardo entre cambio de la secuencia de luces

CONTA1=255

CONTA2=255

Retardo total de la secuencia de luces:

Se asignó una variable (TEMPORIZADOR) con un valor =30 cada vez que se completa la secuencia de luces va decrementando hasta llegar a 0 y de ahí retorna al contador son 11 repeticiones entonces tenemos que:

# CONCLUSIONES

# BIBLIOGRAFIA

[1]. Microcontroladores dsPIC Diseño práctico de

Aplicaciones. Tercera Edición. J. Mª. Angulo

Usategui y I. Angulo Martínez. Editorial McGraw

Hill, 2007

[2]. Microcontroladores PIC Diseño práctico de

Aplicaciones. Tercera Edición. J. Mª. Angulo

Usategui y I. Angulo Martínez. Editorial McGraw

Hill, 1999

[3]. Microcontroladores PIC. La clave del diseño.

E. Martín Cuenca , J. Mª. Angulo Usategui y I.

Angulo Martínez. Editorial Thomson

[4]. Microcontroladores PIC, la solución en un

chip J. Mª. Angulo Usategui, E. Martín Cuenca y

I. Angulo Martínez. Editorial Paraninfo, 2000

[5]. Microcontrolador PIC16F84. Desarrollo de

proyectos. PALACIOS, E.- REMIRO, F. y

LÓPEZ, L.J. Febrero 2004. Rústica y CD-ROM,

648 Págs..