COMPILACIÓN, SIMULACIÓN Y CREACIÓN DE APLICACIONES BÁSICAS CON MICROCONTROLADORES

*Microcontroladores, Ingeniería Electrónica UPTC Sogamoso*

*Lady Camila Preciado Ladino*

e-mail: leidy.preciado01@uptc.edu.co

201311553

*Diana Carolina Medina Roa*

e-mail: diana.medinaroa@uptc.edu.co

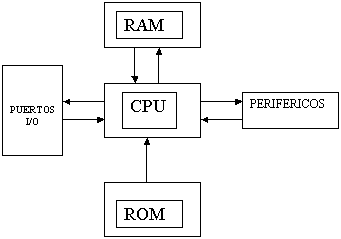
201210175

*Resumen*— *El motivo del presente laboratorio es poder comprender e identificar el funcionamiento y los pasos a seguir para la debida programación del PIC16F887 mediante la aplicación MPLAB X, y comprobar su funcionamiento mediante PROTEUS, realizando las observaciones necesarias, las simulaciones y por último el montaje con sus respectivas pruebas para el buen funcionamiento del dispositivo.*

# Introducción

u

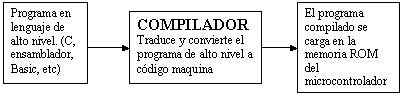
n microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como microcomputadora. Se puede decir con toda propiedad que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado [1].



La representación de datos, instrucciones y señales en forma de bits resulta dificultosa y tediosa para aquellas personas que no estén familiarizadas con el sistema de numeración binario. Aún para los usuarios expertos no resulta tan evidente la interpretación de instrucciones en forma binaria o lenguaje máquina (el lenguaje maquina se le conoce también como lenguaje de bajo nivel debido a que las instrucciones no son propias del lenguaje humano). Es por esto que la programación comúnmente se lleva a cabo en un lenguaje de alto nivel, es decir, un lenguaje que utilice frases o palabras semejantes o propias del lenguaje humano. Las sentencias de los lenguajes de alto nivel facilitan enormemente la programación ya que son familiares a nuestra manera de comunicarnos. Lenguajes como el C o BASIC son comúnmente utilizados en la programación de microcontroladores. Todo programa escrito en un lenguaje de alto nivel debe ser transformado en código máquina. Los programas que escribimos los entendemos nosotros, no así el microcontrolador.

Un software de computadora, llamado compilador, traduce y transforma nuestro programa en código máquina, que es lo que realmente puede leer e interpretar el microcontrolador.

Una vez compilado el programa, es momento de transferir nuestro código maquina hacia la memoria interna del microcontrolador, usualmente hacia la ROM. Para esta tarea se utiliza un programador físico, que es una pieza de hardware que tiene el propósito de escribir el programa en la memoria interna del micro.



.

# Objetivos

* Usar la aplicación de Microchip MPLAB en la creación, compilación, simulación y programación de micro controladores.
* Usar la aplicación PROTEUS como herramienta de depuración y prueba de código realizado en la aplicación MPLAB.
* Implementar aplicaciones sencillas y verificar su funcionamiento en condiciones reales.

# Materiales

* Ordenador con las aplicaciones PROTEUS Y MPLAB X.

Microcontrolador de la serie PIC16F88X.

Programador de microcontroladores PIC.

Protoboard.

Resistencias, Diodos, Switch, pulsadores.

Fuente de alimentación 5V.

# DESCRIPCIÓN DEL LABORATORIO

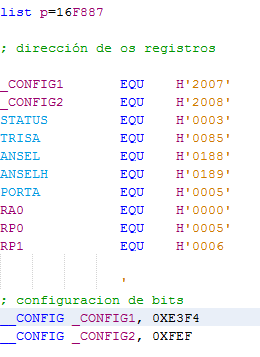
**Contador/conversor (bin a BCD).** Se debe realizar un contador que cuente hasta 255. Cada cambio debe durar mínimo 1 seg. Ésta cuenta se debe visualizar en 12 led. Unidades y decenas serán parte baja y parte alta de PORTB y centenas serán la parte baja de PORTC. Además, deberá contar con una opción de carga de datos, de esta manera se podrá cargar el valor de inicio del respectivo conteo por un dip switch a PORTA, este valor será cargado en base hexadecimal. Cada que el conteo finalicé se deberá visualizar sobre los 12 led una determinada secuencia de intermitencia diseñada por el estudiante que duré no menos de 1 min. Debe contar con una opción de reinicio.

# Procedimiento

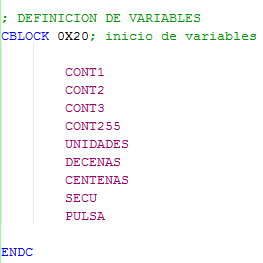
De acuerdo a la guía del laboratorio se siguieron los pasos para crear el proyecto con el programa de simulación MPLAB X, identificando las funciones de dicho programa, además las características del microcontrolador Pic16F887 que vienen dadas en el datasheet del dispositivo.

Después de reconocer el entorno de trabajo, se empezó con el proyecto Contador/conversor (BIN a BCD).

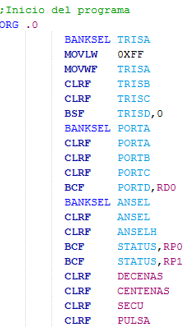
Se definió el PIC y los registros con su respectiva dirección, configurando los CONFIG1 Y CONFIG2.



Las variables o registros de propósitos generales se definieron a partir de la posición 0X20. LAS CONT1, CONT2, CONT3, son para los retardos. UNIDADES, DECENAS, CENTENAS para el conversor binario a BCD. SECU para la secuencia y PULSA para el RESET.



Se define el inicio del programa:

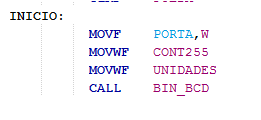


El BANKSEL selecciona el banco donde se va a trabajar, estos corresponden al RP0 y RP1 del registro STATUS, en este casi sele orden que seleccione el banco donde esta TRISA, en este se encuentran todos los TRIS, MOVLW hacemos que el literal se ponga en alto es decir 0XFF en notación hexadecimal y se guarde en W, con MVWF se dice que lo que está guardado en W se mueva a F es decir a TRISA es decir se ponga en alto, con esto decimos que PORTA va a ser una entrada.

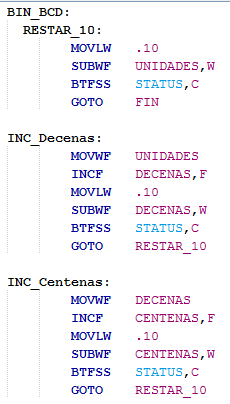
Para no tener problemas a la hora de ejecutar el programa con el comando CLRF borramos PORTA, PORTB, PORTC, para ponerlos como salidas.

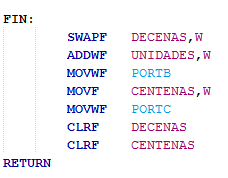
Mediante el comando BSF se asigna el bit 0 de PORTD como un alto. BANKSEL llama los ANSEL Y CLRF borra el contenido de ellos, después escogemos el banco donde se encontrarán nuestras variables creadas como DECENAS, CENTENAS... que será el banco 0 el cual le asignamos clareando RP0 Y RP1; esta es otra manera de asignar los bancos ya sea poniendo en 0 o 1 RP0 o RP1 ya sea el caso de vez de hacer el uso de BANKSEL ya que este siempre se demorara 2 Tcy.

Y por último se vuelve a borrar los datos que puedan contener las variables creadas.



Definimos como primer paso MOVF es decir PORTA a W y con MOVWF guardamos el dato que se encuentre en W en las variables CONT255 y UNIDADES. Después de hacer esto llamamos la conversión BIN\_BCD.





**BIN\_BCD** se realiza la conversión de binario a BCD.

Para realizar dicha conversión se tiene una serie de pasos:

**RESTAR\_10:**

MOVLW el literal se mueve a W .10 (en notación decimal), SUBWF se resta a UNIDADES lo que está en W es decir 10. BTFSS si la bandera C es decir el carry está en 1 omita y vaya a la siguiente instrucción en este caso GOTO FIN, es decir vaya a la siguiente instrucción FIN.

**INC\_DECENAS:**

MOVWF se mueve lo que está en W a F es decir UNIDADES. INCF se incrementa la variable DECENAS. MOVLW se mueve el .10 (en notación decimal).

SUBWF restamos a F lo que se encuentra en W, es decir a DECENAS restamos 10. BTFSS si la bandera C es decir el carry está en 1 omita y vaya a la siguiente instrucción en este caso RESTAR\_10.

**INC\_CENTENAS:**

MOVWF se mueve DECENAS a W. INCF la variable CENTENAS. MOVLW se mueve el .10 (en notación decimal).

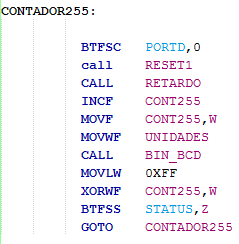
Igual que en INC\_DECENAS se hace el mimo procedimiento, SUBWF restamos a F lo que se encuentra en W, es decir a CENTENAS restamos 10. BTFSS si la bandera C es decir el carry está en 1 omita y vaya a la siguiente instrucción en este caso RESTAR\_10.

**FIN:**

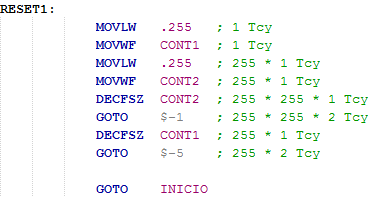
SWAPF esta instrucción intercambia los bits de mayor peso con los de menor peso, es decir los 4 bits de menos peso de la variable DECENAS se mostrará en los bits del 7 al 4 del registro W ,los bits del 7 a 4 de DECENAS pasaran a los bits 0 a 3 del registro W (ya que solo se manejan en unidades los 4 primeros bits estos se encontrarán en cero) , después con la operación ADDWF, esta instrucción añade lo que hay en UNIDADES en W, ya que tenemos solo datos en los bits 7 a 4 de W (DECENAS), se añade el valor de UNIDADES y como este valor se encuentra de los bits 3 a 0 no afectaría el dato de decenas; esto se visualizara en PORTB. CENTENAS se mostrará en PORTC. Nuevamente con CLRF se limpian las variables DECENAS Y CENTENAS, por último, se retorna al programa principal.

Para PORTB queda en el siguiente orden: del bit 7 al 4 se mostrará el valor de DECENAS y del bit 3 al 0 se visualizará UNIDADES

* El siguiente paso a seguir es teniendo el dato guardado en la variable UNIDADES empezar a contar hasta 255, teniendo en cuenta el RESET si está en 1 significa que se ha oprimido, si no, su estado normal será 0.



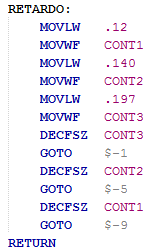
El BTFSC significa omita si está en 0 si el bit del PORTD está en 0, es decir el push button estará en su estado normal, de lo contrario con el CALL llamamos el RESET1 el cual es un retardo que dura 196,352 ms, este retardo de hace por el ruido eléctrico que emite el push button.



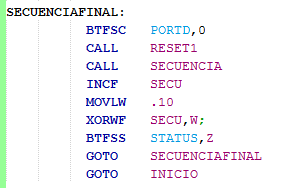
Este retardo se hace con el fin de evitar rebote, ya que el push button no es ideal y al momento de tener el RESET en alto presentaría unas pequeñas variaciones en la práctica que confundirían al PIC y así el circuito funcione sin problemas.

**NOTA: se hace llamado a la función RESET en el inicio de CONTADOR255 y en SECUENCIAFINAL, después de preguntar que si nuestra entrada del bit 0 de PORTD esa activado o no.**

Si el RESET está en 0 este paso se omite y con el CALL se llama RETARDO que dura 1 minuto.

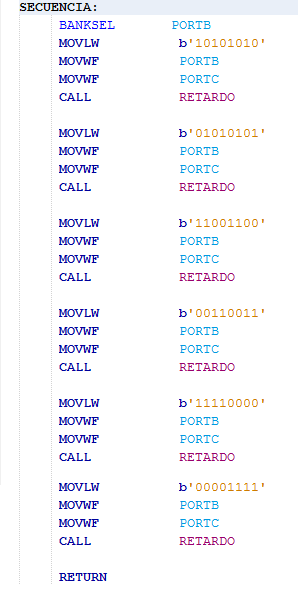


Después del llamado de retardo este retorna y en este momento comienza a aumentar nuestra variable CONT255, manda este valor a UNIDADES que como se aclaró anteriormente es la encargada de pasar el dato de binario a bcd, hace llamado a BIN\_BCD; le cargamos un literal a W que en este caso sería nuestro máximo de conteo (255), el cual lo comparara con nuestra variable CONT255 mediante una XOR, cuando estos valores sean iguales se encenderá nuestra bandera Z (todos ceros ) la cual nos indica que ya l ser igual termino el conteo y pasara a nuestra siguiente función que será :



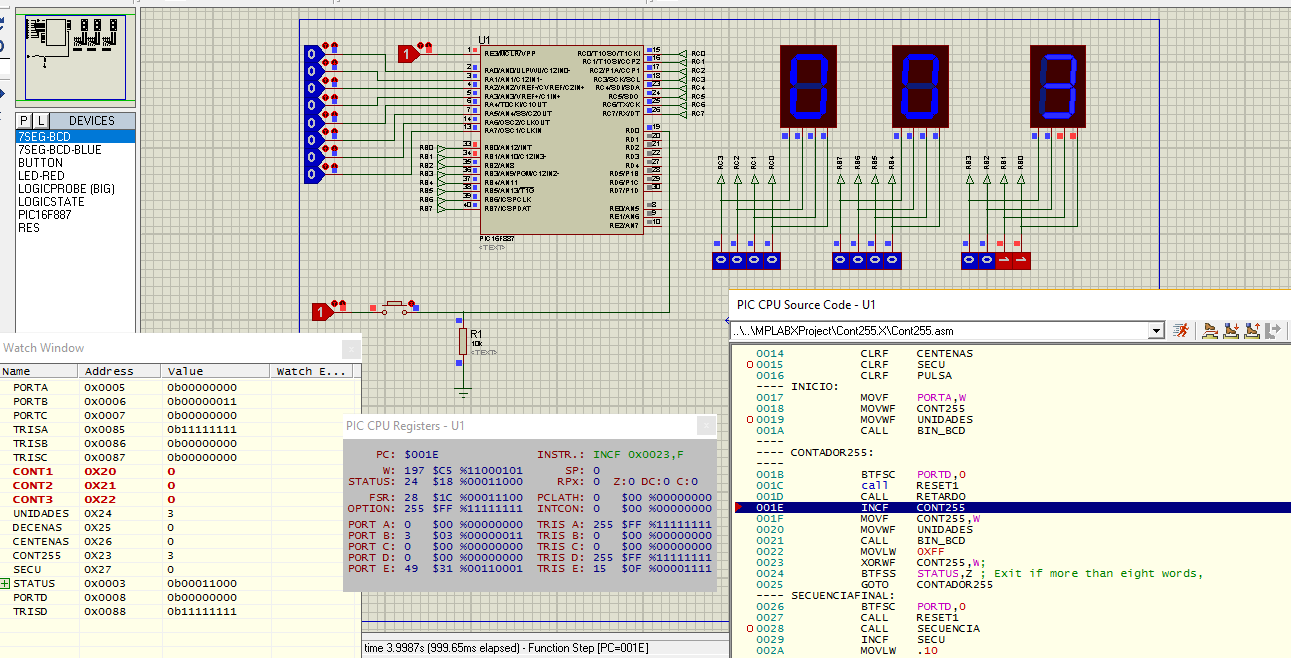
En esta parte muestra la secuencia que harán los LEDS después de terminar su conteo hasta 255, en la primera parte vemos que se encuentra la pregunta del RESET como se aclaró en la nota,

después de esto no hace llamado a SECUENCIA

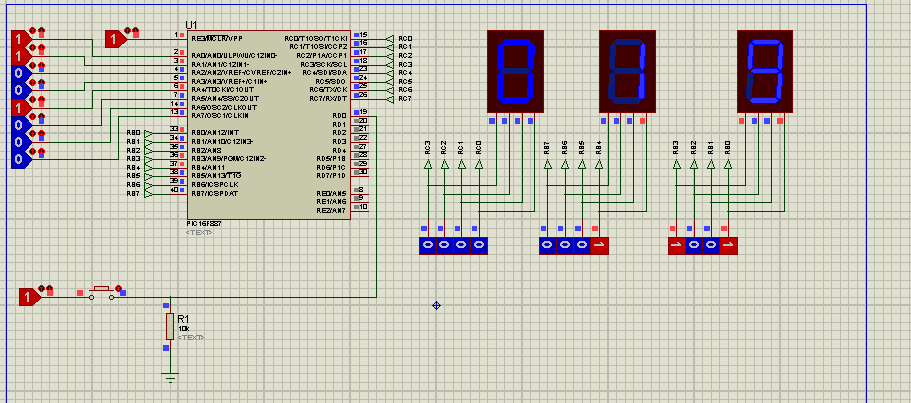


Se observa que se inserta un valor en binario a PORTB y PORTC los cuales son las salidas hacia los LED, entre cada cambio de secuencia se llama a RETARDO que en este caso es el mismo de 1 Seg que se calculó para el conteo. Después de finalizar este ciclo se retorna a SECUENCIAFINAL, donde incrementa una variable llamada SECU, esta lleva el conteo de las repeticiones que hace secuencia, como secuencia dura con todos los retardos 6 Seg se necesita que se repita 10 veces para el total dure 60 Seg, al igual que en contador se guarda en W el valor de 10 y lo se compara mediante una XOR con SECU el cual cuando este sea igual se activara la bandera Z (ceros ), dando así por terminada la instrucción y devolviéndose a INICIO para retornar a tomar el valor que hay en PORTA e inicializar todo el programa de nuevo.

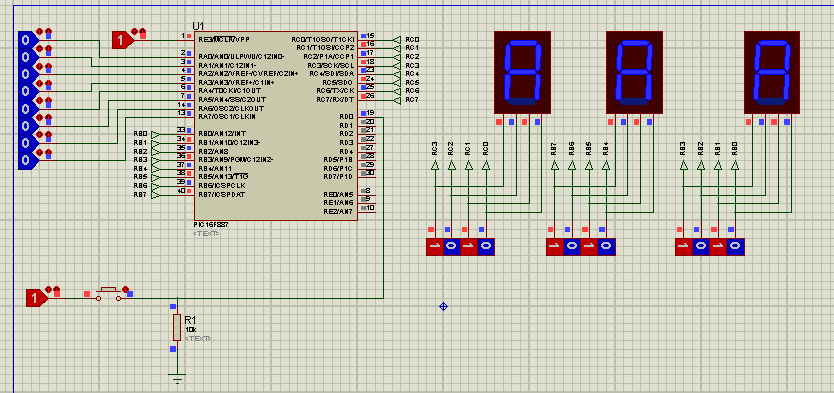
Para comprobar si el programa está funcionando como se espera se manejará PROTEUS cargando el programa al PIC, para la verificación de este se ejecutó paso a paso, visualizando que las ordenes se ejecutaran como se deseaba.

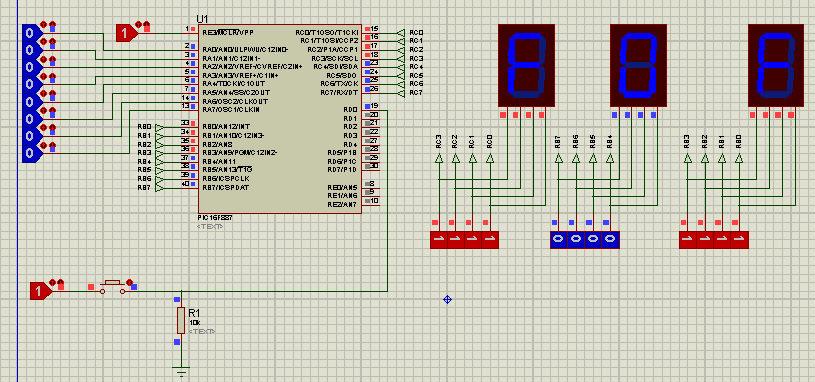


Y finalmente ya al comprobar el funcionamiento se simulará correctamente en PROTEUS.



Visualización de algunas partes de la secuencia





**NOTA: El diagrama de flujo del programa se encuentra dentro de los anexos**

# CONCLUSIONES

* Se logró identificar el entorno de trabajo, desde el software de simulación MPLAB X hasta el PIC, por medio del datsheet del dispositivo, también los comandos necesarios para que el programa se pueda ejecutar sin presentar algún problema.
* Se observó que la programación del PIC presenta cierta complejidad por esto la mejor guía es el datasheet que el dispositivo brinda a los usuarios.
* Para poder saber si el programa funciona correctamente es necesario ir probando mediante otro simulador que sirva de ayuda en este caso PROTEUS.
* Se presentaron problemas al pasar el programa al dispositivo por medio del PICKIT 3, ya que pedía que tenía que estar alimentado con una fuente externa y al hacer esto se presentó el mismo problema, por lo cual fue necesario buscar otro computador y otro programador
* Se presentó ruido eléctrico en la parte del RESSET ya que el push button no mantiene un pulso ideal provocando un rebote, por esta razón fue necesario añadirle un retardo para que no afectara el programa.

# REFERENCIAS Y BIBILIOGRAFIA

[1]http://www.electronicaestudio.com/microcontrolador.htm

[2] http://hyperphysics.phy-astr.gsu.edu/hbasees/audio/amp.html

[3] https://es.wikipedia.org/wiki/Microcontrolador

[4] http://ww1.microchip.com/downloads/en/DeviceDoc/41291D.pdf

[5] https://github.com/maurinc2010/Curso\_uC

.