Compilación, Simulación y Creación de Aplicaciones Básicas con Microcontroladores (Marzo 2017)

Ingrid Catherin Morales, Alejandro Antonio Nuñez, *U.P.T.C*

***Resumen*— En esta guía se presenta el diseño de un contador de código BCD, para el cual se usó un contador binario y un convertidor de código binario a código BCD. Se describe cada una de las funciones utilizadas justificando su elección. También se muestran los resultados obtenidos con la implementación.**

***Índice de Términos*— Código binario, contador BCD, convertidor.**

I. Introducción

Los microcontroladores pueden encontrarse día a día en la mayoría de dispositivos electrónicos usados en la actualidad, desde teléfonos celulares, reproductores de música, hasta en aplicaciones industriales y medios de transporte masivos.

Con el continuo desarrollo de la tecnología en la historia se llegó a pasar de computadoras que ocupaban incluso plantas completas de infraestructura, a pequeñas computadoras portables de apenas unos milímetros. Es allí donde surge la gran importancia y versatilidad de los microcontroladores como dispositivos inteligentes de pequeño tamaño y costos moderados, que pueden llegar a cumplir con innumerables funciones en diversos circuitos.

Por todo lo anteriormente mencionado, el estudio, caracterización y uso de los microcontroladores se convierte en un pilar para los ingenieros como solución a múltiples problemas y desarrollo de nuevas tecnologías.

II. Materiales y equipos

- Ordenador con las aplicaciones PROTEUS Y MPLAB X.
- Microcontrolador de la serie PIC16F88X.
- Programador de microcontroladores PIC.
- Protoboard.
- Resistencias, Switch, pulsador, diodos LED, alambre.
- Fuente de alimentación 5V
- PC y Software de simulación.

III. Procedimiento

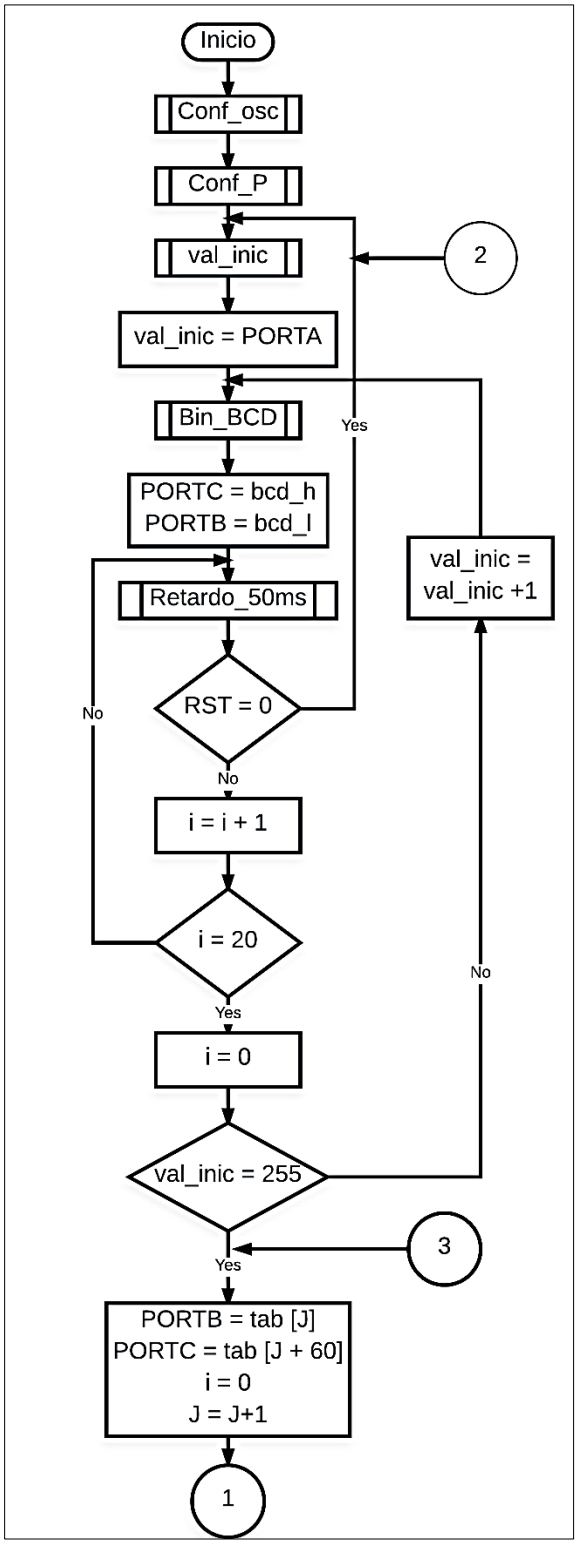
Contador/conversor (bin a BCD). Se debe realizar un contador que cuente hasta 255. Cada cambio debe durar mínimo 1 segundo. Ésta cuenta se debe visualizar en 12 led. Unidades y decenas serán parte baja y parte alta de PORTB y centenas serán la parte baja de PORTC. Además, deberá contar con una opción 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 hexa. Cada que el conteo finalice, 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.

IV. Desarrollo de la práctica

Diagramas de Flujo

El programa utilizado consiste en un contador binario básico y un convertidor de código binario a BCD. Para el contador Binario se utilizan las funciones básicas de incremento en los registros y el convertidor se establece por medio de restas sucesivas.

En fig. 1 se observa el diagrama de flujo del programa principal.



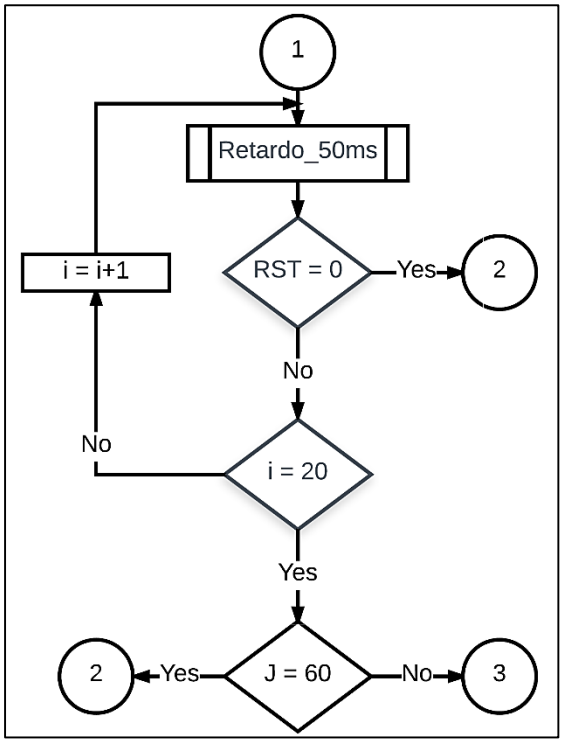


Fig. 1. Diagramas de flujo del programa principal.

Se hace uso de varias rutinas para hacer simplificar el diagrama principal. La rutina Conf_osc cumple la función de configurar el oscilador, en ella se configura un oscilador interno para frecuencia de 8Mhz. El diagrama de flujo de esta rutina está en fig. 2.

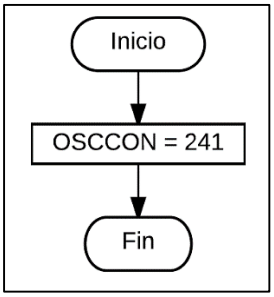


Fig. 2. Diagrama de flujo de la subrutina Conf_osc

La siguiente subrutina utilizada Configura los puertos (Rutina Conf_P), es decir, el puerto A como entrada (Valor de inicio del conteo), Puerto B y C como salidas del contador y un pin del puerto E para reinicio; además se configuraron las posibles entradas ADC como pines digitales y se inicializan los puertos B y C. El diagrama de flujo de la rutina se encuentra en Fig. 3.

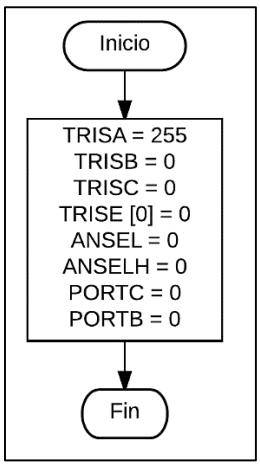


Fig. 3. Diagrama de flujo de la rutina de configuración de puertos.

Otra rutina utilizada inicializa todas las variables con un valor de 0 (val_inic), su diagrama de flujo está en fig. 4.

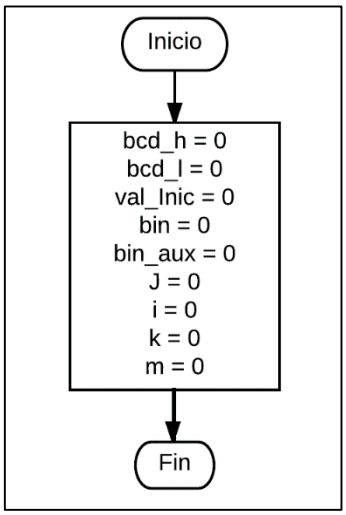


Fig. 4. Diagrama de flujo de inicialización de variables.

El convertidor se hizo como una rutina. Se ingresa el valor a convertir en la variable val_inic y el resultado se guarda en bcd (variable de 16 bits). La rutina resta centenas, decenas y unidades en el orden indicado, si el resultado es positivo se suma el valor correspondiente al número restado, es decir si se resta 100 y el resultado es positivo, a la variable bcd se suma el número 100_h. El diagrama de flujo de esta rutina (Bin_BCD) se observa en fig. 5.

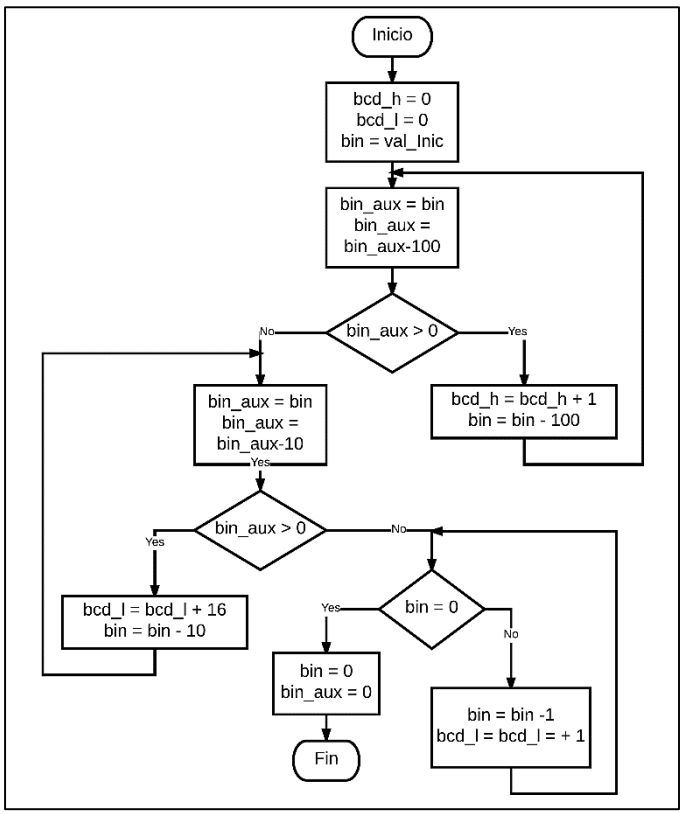


Fig. 5. Diagrama de flujo del convertidor de código binario a BCD utilizado.

La última rutina utilizada es la correspondiente al retardo por software, en ella el microcontrolador tarda aproximadamente 50 milisegundos (ms). Para esta se tuvo en cuenta que la función decfsz (decremento de un registro F, saltar si el resultado es 0) cuando hace el decremento del valor 0 se desborda (su siguiente valor es 255) y dado que el resultado no es 0, simula contar desde 256 hasta 0. Se calcula la ecuación que describe el tiempo de ejecución de las instrucciones:

$$t = 2\,T_{cy} + [256 * 3 + 2]m\,T_{cy} \quad (1)$$

El valor de m es,

$$m = 130$$

Esta rutina se llamó Retardo_50ms y se encuentra en fig. 6.

Este tiempo se elige para aparentar una respuesta “inmediata” del sistema, dado que el ser humano no le es fácil detectar un retardo de un tiempo tan pequeño. Para cumplir con el tiempo de visualización de un segundo se esperan 20 rutinas de retardo

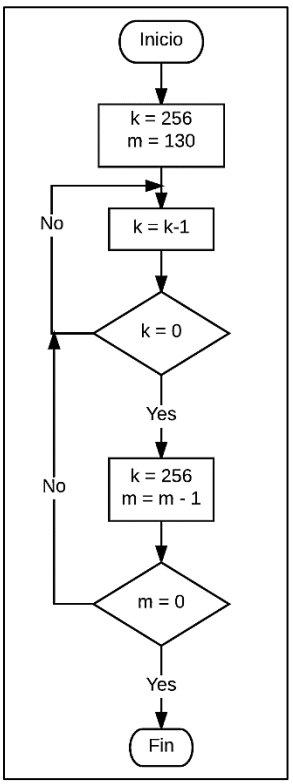


Fig 6. Diagrama de flujo del retardo utilizado.

Para la rutina de intermitencia se usa una tabla de 60 posiciones con 12 bits; se consigue seccionando la tabla en porciones de 8 bits y desechando los 4 bits más significativos de la parte alta. La parte baja ocupa las primeras 60 posiciones y la parte alta se encuentra en las siguientes 60 posiciones. Se elige 1 segundo para visualizar cada posición de la tabla.

En el programa principal se cumplen varias funciones:

- Contador binario.
- Se evalúa cada 50 ms si se ha pulsado el reset y si es así reinicia el programa parcialmente, es decir reinicia el programa contando desde el valor del puerto A
- Evaluar si ya se desbordo el contador e iniciar la rutina de intermitencia.
- Dirigir la ejecución de rutinas.
- Contabilizar el tiempo de la visualización de los datos en los puertos.

Tabla de Secuencias

A continuación, se muestra la tabla correspondiente a la secuencia de LED diseñada, donde cada posición de la tabla dura 1 segundo.

TABLA I. SECUENCIA DE INTERMITENCIA DISEÑADA

	PORTC				PORTB							
seg	3	2	1	0	7	6	5	4	3	2	1	0
1	■	■										
2		■	■									
3			■	■								
4				■	■							
5					■	■						
6						■	■					
7							■	■				
8								■	■			
9									■	■		
10										■	■	
11											■	■
12										■	■	
13									■	■		
14								■	■			
15							■	■				
16						■	■					
17					■	■						
18				■	■							
19			■	■								
20		■	■									
21	■	■										
22	■											
23		■										
24				■								
25						■						
26								■				
27									■			
28												■
29									■			
30								■				
31						■						
32				■								
33		■										
34					■	■	■	■				
35	■	■	■	■					■	■	■	■
36					■	■	■	■				
37	■	■	■	■					■	■	■	■
38					■	■	■	■				
39	■	■	■	■					■	■	■	■
40					■	■	■	■				
41	■	■	■	■					■	■	■	■
42					■	■	■	■				
43	■	■			■	■			■	■		
44			■	■			■	■			■	■
45	■	■			■	■			■	■		
46			■	■			■	■			■	■

47	■	■			■	■			■	■		
48			■	■			■	■			■	■
49	■	■			■	■			■	■		
50			■	■			■	■			■	■
51	■	■			■	■			■	■		
52		■		■		■		■		■		■
53	■		■		■		■		■		■	
54		■		■		■		■		■		■
55	■		■		■		■		■		■	
56		■		■		■		■		■		■
57	■		■		■		■		■		■	
58		■		■		■		■		■		■
59	■		■		■		■		■		■	
60		■		■		■		■		■		■

A partir de la Tabla 1 se obtienen los valores equivalentes en decimal para cada uno de los puertos en uso (PORTB y PORTC), para ser ingresados en el programa con ayuda del comando DT.

Simulación

Uno vez terminado el programa en MPLAB X® se procede a realizar la simulación en Proteus ISIS, como se muestra en la Fig.7.

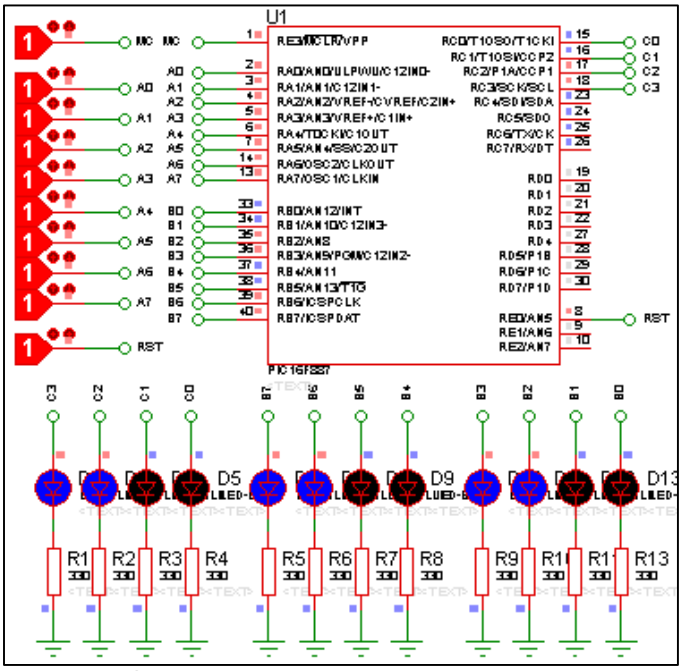


Fig. 7. Esquemático en Proteus del contador BCD.

Implementación

Una vez se verificó que la simulación funcionaba correctamente, se prosiguió a implementar en Protoboard el correspondiente circuito, teniendo en cuenta las siguientes consideraciones:

- Corriente máxima por puerto del PIC: 25 mA
- Voltaje de alimentación: 5V
- Diferencia de potencial de los diodos:
  - Rojo: 1,8 a 2,2 V
  - Anaranjado: 2,1 a 2,2 V
  - Amarillo: 2,1 a 2,4 V
  - Verde: 2 a 3,5 V
  - Azul: 3,5 a 3,8 V
  - Blanco:3,6 V

Teniendo en cuenta los datos nombrados se tiene en cuenta el circuito correspondiente a la conexión de los diodos LED para determinar un valor de resistencia apropiado para evitar daños en el integrado. Por tanto,

$$R = \frac{5V - 1{,}8V}{25mA} = 128\ \Omega \qquad (2)$$

Lo que indica que cualquier valor de resistencia superior a los 128 Ω garantizaría un buen nivel de corriente, por lo cual, se eligió una resistencia de 330 Ω, tanto para diodos LED como para dip switch y pulsadores, siendo este un valor de resistencia prudente para el correcto funcionamiento del circuito.

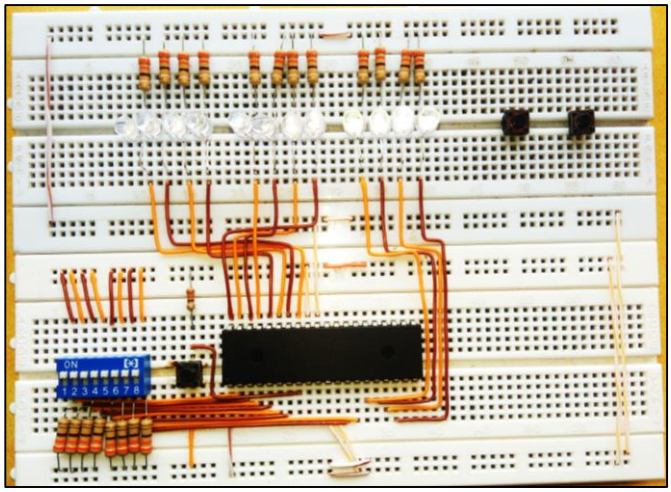


Fig. 8. Montaje en Protoboard del Contador BCD.

La Fig.8. muestra el montaje final para el contador BCD en funcionamiento.

V. Conclusiones

Antes de comenzar el desarrollo del programa en MPLAB X, se debe tener en cuenta las recomendaciones y el set de instrucciones de la hoja de especificaciones, con el fin de evitar una extensión innecesaria del código y posibles daños en el integrado.

La utilización de retardos por software no es una forma óptima de utilización del microcontrolador porque durante

mucho tiempo se encuentra haciendo operaciones innecesarias, este tiempo podría ser invertido en otras funciones como disminuyendo el uso de energía y procesador, mejorando la eficiencia, esto se puede lograr con el uso de interrupciones y temporizadores (Timers).

A la hora de la implementación del circuito, es de vital importancia un buen orden y cableado, para evitar posibles errores, debido a que los montajes en el área suelen ser bastante engorrosos y extensos.

Referencias

[1] Harprit Singh Sandhu. Making Pic Microcontroller Instruments and Controllers. McGraw-Hill,2009.
[2] Randal Hyde. The Art of Asembly Languaje. 2nd Edition. No starch press, 2010.
[3] http://www.microchip.com