LABORATORIO N° 1 COMPILACION, SIMULACION Y CREACION DE APLICACIONES BASICAS CON MICROCONTROLADORES

Steven Velandia Vargas 201020985

***Abstract*—This paper presents the development of some applications based on microcontrollers, specifically the PIC16F887. An incremental counter was made from 0 to 255, followed by a sequence with a duration of one minute, with the option to restart the application at any time.**

***Index Terms*—Assembler, Microcontroller, Programmer.**

I. INTRODUCCION

EN el desarrollo de aplicaciones con microcontroladores son de vital importancia las herramientas que permiten llevar a cabo esta labor, como el MPLAB y el PROTEUS, los cuales son sólo una pequeña muestra de los software desarrollados con este fin. Con el uso de los mismos se pueden realizar las operaciones de creación, depuración y terminado final de sistemas de control completo realizadas alrededor del uso de estos versátiles dispositivos.

Son muchas las aplicaciones que se pueden llevar a cabo con los microcontroladores, tareas sencillas hasta tareas que pueden llegar a ser mucho más complejas. Es debido a esto que el uso de los microcontroladores se hace cada vez más necesario y se puede ver inmerso en casi todas las actividades que realizamos a diario con nuestros equipos electrónicos. El uso de un control remoto de una TV, los celulares, las luces de los automóviles, el sistema de vidrios eléctricos, parabrisas, etc. Un microcontrolador interviene en todas estas aplicaciones y muchas más que puedan requerir un control.

II. OBJETIVOS

- ✓ Usar la aplicación de Microchip MPLAB en la creación, compilación, simulación y programación de microcontroladores.
- ✓ 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.

III. MATERIALES Y EQUIPOS

- Ordenador con las aplicaciones PROTEUS y MPLAB.
- Microcontrolador de la serie PIC16F887.
- Programador de microcontroladores PIC.
- Protoboard.
- Resistencias, diodos, switch, pulsadores, etc.
- Fuente de alimentación 5V.

IV. MARCO TEÓRICO

La diferencia entre microprocesador y microcontrolador radica en que, al microcontrolador se le diseña de tal manera que tenga todos los componentes integrados en el mismo chip. No necesita de otros componentes especializados para su aplicación, porque todos los circuitos necesarios, que de otra manera corresponden a los periféricos, ya se encuentran incorporados. Así se ahorra tiempo y espacio necesario para construir un dispositivo.

Ahora, 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 microcomputador. Se puede decir con toda propiedad que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado.

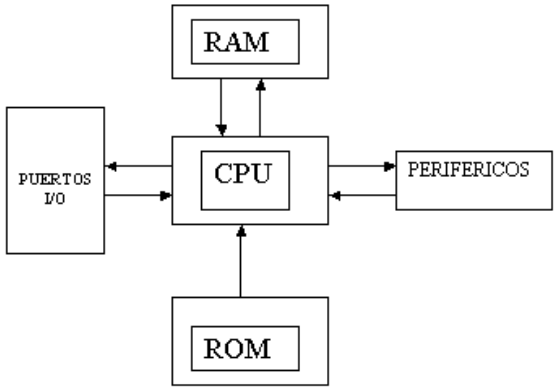


Figura 1. Diagrama de un microcontrolador.

Los microcontroladores están diseñados para interpretar y procesar datos e instrucciones en forma binaria. Patrones de 1's y 0's conforman el lenguaje máquina de los microcontroladores, y es lo único que son capaces de entender. Estos 1's y 0's representan la unidad mínima de información, conocida como bit, ya que solo puede adoptar uno de dos valores posibles: 0 o 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. Aun 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). Un tipo de lenguaje más especializado es el lenguaje ensamblador, el cual es una lista con un limitado número de instrucciones a los cuales puede responder un microcontrolador. Estas instrucciones son palabras o abreviaciones que representan las instrucciones en lenguaje máquina del microcontrolador.

Las instrucciones en lenguaje ensamblador, también conocidas como nemotécnicos, son fáciles de entender y permiten operar directamente con los registros de memoria así como con las instrucciones intrínsecas del microcontrolador. Es por esto que el lenguaje ensamblador es sin lugar a dudas el lenguaje por excelencia en la programación de microcontroladores, ya que permite hacer un uso eficiente de la memoria y minimizar el tiempo de ejecución de un programa.

Un software de computadora, llamado compilador, traduce y transforma nuestro programa en código máquina. Una vez compilado, se transfiere el código hacia la memoria interna del microcontrolador (usualmente 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. [1]

Figura 2. Diagrama para ejecutar un programa en un micro.

V. PROCEDIMIENTO

Contador/conversor (bin a BCD). Se debe realizar un contador que cuente hasta 255. Cada cambio debe durar mínimo 1seg. Esta cuenta se debe visualizar en 12 LED. Unidades y Decenas serán la parte baja y parte alta del 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 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 dure no menos de 1 min. Debe contar con una opción de reinicio.

Con base en el ejercicio propuesto se procedió a realizar los diagramas de flujo correspondientes al programa y con base en estos poder realizar la programación del mismo. Los diagramas de flujo se encuentran en los Anexos.

El programa realizado fue hecho con varias subrutinas, que luego serían unidas o llamadas en un programa principal llamado Main. Las subrutinas se hicieron de la siguiente forma:

CONFIGURACIONES

Lo primero que se hizo fue la configuración del oscilador que se iba a utilizar, por medio del Datasheet del PIC se verificó el valor que debía tener el registro OSCCON, el cual está dispuesto para la asignación de la frecuencia interna del microcontrolador que queremos usar, para esta práctica se utilizó una frecuencia de oscilación de 125KHz.

A continuación se hizo la configuración de los puertos que se debían disponer como entradas y salidas, en este caso los puertos B y C iban a ser salidas, así que se configuró el registro TRISX para cada puerto poniéndolo en 0 para asignar a estos como salida. A los puertos que se les asignó como entrada fueron los puertos A y el bit 0 del puerto D, debido a que por medio del puerto A se asignará un valor para ingresar al contador y por medio del puerto D se hará el RESET del programa. Así mismo se configuraron los registros ANSEL y ANSELH poniéndolos en 0, para que la entrada por los puertos A y D fuera de tipo digital.

RETARDO

En esta subrutina se configuraron dos retardos, uno para el conteo de 0 a 255 de 1 seg entre cada número y un retardo de 0.5 seg para la secuencia.

Para el retardo del conteo se procedió a crear dos variables, puesto que como 255 es el máximo número que tendremos. Se le puede asignar un valor aleatorio a una de las variables y de la ecuación obtenida del contador, se despeja el valor de la otra variable para que nos dé el valor de la frecuencia que se necesita, en este caso de 1 seg entre cada cambio de número.

$$t = \frac{(2 + 3X + 3XY) * 4}{Fosc}$$

Con base en la ecuación, a X se le asignó un valor de 100, y como se había dicho la frecuencia de oscilación será de 125KHz, así que se determina el valor de Y, obteniendo 102, como se verá en el código.

$$1Seg = \frac{(2 + 3(100) + 3(100)Y) * 4}{125K}$$

Y =102

Cuando se empiece a decrementar el valor de esta segunda variable que será en el programa el contador número 2, si éste llega a ser 0, saltará a decrementar el contador número 1, de lo

contrario regresará a seguir decrementando el valor del contador 2, y del mismo modo se hará cuando el contador 1 llegue a ser 0 o no.

Para el retardo 2, utilizado en la secuencia, se procedió a hacer el mismo programa del retardo 1, pero para un tiempo de 0.5 seg entre cada cambio de la secuencia, por lo cual se halló un valor para el contador 2 de 51, la mitad del contador anterior.

CONVERSOR

Para esta subrutina, se creó una variable nueva (X), a la cual será asignado el valor del contador 1, luego se incrementará el registro de las unidades y se decrementará el valor de X, de tal modo que si X llega a ser 0 retornará la subrutina a donde ha sido llamada para seguir con el programa pero si X aún no es 0, incrementará el valor de las unidades. Luego se asigna un valor de 10 al registro de trabajo (W) y se compara con el registro de las unidades (UNI). Se hace un test del bit 2 del registro STATUS, el cual corresponde a la bandera de Z (cero). Si este bit está encendido quiere decir que se clarearán las unidades y se incrementarán las decenas, si no, regresará a seguir decrementando el valor de X.

A continuación se hace lo mismo con las decenas, se le asigna un valor de 10 al registro de trabajo y se compara con el valor de las decenas (DEC), haciendo el mismo test del bit 2 del registro STATUS y prosiguiendo a que si este bit está encendido (1), se clarearán las unidades y decenas y se incrementará el registro de las centenas, de lo contrario regresará a seguir decrementando el valor de X.

Para las centenas se hace el mismo procedimiento que para las unidades y las decenas, asignando un valor de 10 al registro de trabajo y comparándolo con el registro de las centenas (CEN), se testea el bit 2 del registro STATUS y si está en 1 se clarearán los registros de unidades, decenas y centenas, retornando al programa principal donde fue llamado. De lo contrario regresará a seguir decrementando el valor de X.

VER

En esta subrutina, se asigna el valor del registro de las decenas (DEC) a una variable nueva (A), la cual será asignada al nibble alto del puerto B, para hacer esto, se hace uso de la instrucción SWAPF, la cual permite poner en el nibble alto el valor del registro A, en este caso las decenas. Luego se mueve el valor del registro de las unidades al registro de trabajo (W) y se hace una OR inclusiva, la cual permite que se sumen los valores del registro A y W (trabajo); procediendo a asignar este valor al puerto B. Por último, se asigna el valor del registro de las centenas al puerto C y se clarean los registros de las unidades, decenas y centenas. Retornando luego al lugar donde fue llamada esta subrutina en el programa principal.

SECUENCIA

Se realizó una secuencia que tiene una duración de 6 segundos, lo cual no es suficiente para la duración requerida que es de 1 min. Así que se le asignó un valor de 10 a una variable llamada MIN, la cual será el registro que lleve la cuenta hasta que la secuencia se halla repetido 10 veces, dando un tiempo de 60 seg = 1 min.

La secuencia se hizo asignando valores a los puertos B y C y llamando a la subrutina de RETARDO2, la cual es de 0.5 seg. Se testea el valor del bit 0 del Puerto D y si está en 0, sigue realizando la secuencia, de lo contrario regresará al principio del programa principal. Lo mismo se hace con cada valor asignado a los puertos y luego se testea el bit 0 del puerto D para saber si el programa ha sido reseteado o no.

Al finalizar el primer ciclo de la secuencia, se va decrementando la variable MIN, de tal modo que si esta llega a ser cero, retornará al lugar donde fue llamada esta subrutina en el programa principal, y si no entonces regresará a empezar la secuencia de nuevo.

MAIN

Esta es la subrutina principal, en la cual se llamarán todas las demás subrutinas y en la cual, lo primero que se hace es llamar la configuración del PIC16F887, luego se asigna el valor de la posición de los registros a utilizar en este programa y los bits de algunos de estos registros, como el RA0, RA1, RP0 Y RP1.

A continuación se declaran todas las variables a utilizar en el programa a partir de la posición 0X20 del banco 0, debido a que desde esta posición están asignados los valores de los registros de propósito general.

Se procede a llamar a la subrutina de las configuraciones del oscilador y de los puertos y se hace una etiqueta llamada LOOP, con la cual se da inicio al programa principal.

En la etiqueta LOOP se comienza incrementando el valor del contador 1 para realizar el conteo de 0 a 255, luego se llama a la rutina CONVERSOR, VER y RETARDO, se procede a hacer un test del bit 0 del puerto D para saber si se ha reseteado el programa y si este bit está encendido (1), el valor del puerto A será asignado al contador 1 y regresará al conversor para volver a hacer el ciclo y poder mostrar el valor asignado en los LED, si no es 1, regresará al conversor. Después de esto se le asigna un valor de 0XFF (255) al registro de trabajo y se compara este valor con el valor del contador 1 para saber si son iguales. Se testea el bit 2 del registro STATUS para saber si este resultado ha dado 0 y si no, regresará a la etiqueta LOOP, si sí entonces pasará a la secuencia. Debajo del ciclo LOOP, se incluirán las subrutinas creadas y se finalizará el programa.

VI. CONCLUSIONES

- Se comprobaron los conocimientos adquiridos sobre el manejo de las configuraciones de los puertos para asignarlos como entradas o salidas, al igual que el uso de retardos para situaciones reales.
- Se adquirieron nuevos conocimientos sobre la programación de un microcontrolador por medio de la herramienta de programación PICKIT, aprendiendo la respectiva conexión para poder escribir el programa realizado en el microcontrolador.
- Se aprendió sobre el uso de nuevas instrucciones en lenguaje ensamblador como las que se utilizaron para realizar el cambio de los nibble alto y bajo de los registros, facilitando así la obtención de los datos a la salida de los puertos del microcontrolador.

VII. REFERENCIAS Y BIBLIOGRAFÍA

[1]. Consultado de:

http://www.electronicaestudio.com/microcontrolador.htm

[2]. Microcontroladores dsPIC Diseño practico de aplicaciones. Tercera Edición. J. Mª. Angulo Usategui y I. Angulo Martínez. Editorial McGraw Hill, 2007

[3]. Microcontroladores PIC Diseño práctico de aplicaciones. Tercera Edición. J. Mª. Angulo Usategui y I. Angulo Martínez. Editorial McGraw Hill, 1999

[4]. Microcontroladores PIC. La clave del diseño. E. Martín Cuenca, J. Mª. Angulo Usategui y I. Angulo Martínez. Editorial Thomson

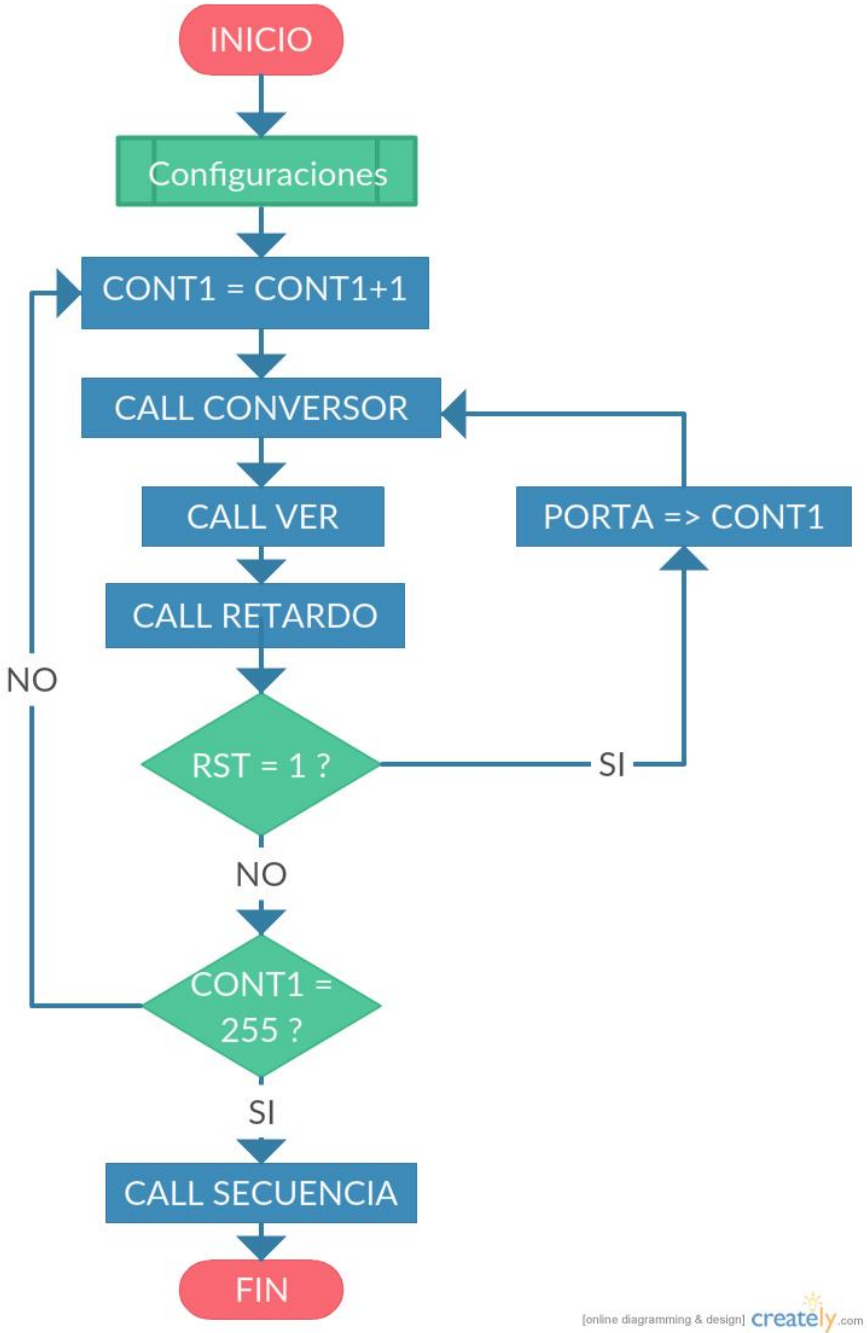
[5]. Microcontroladores PIC, la solución en un chip J. Mª. Angulo Usategui, E. Martín Cuenca y I. Angulo Martínez. Editorial Paraninfo, 2000

[6]. Microcontrolador PIC16F84. Desarrollo de proyectos. PALACIOS, E.- REMIRO, F. y LÓPEZ, L.J. Febrero 2004. Rústica y CD-ROM, 648 Págs..

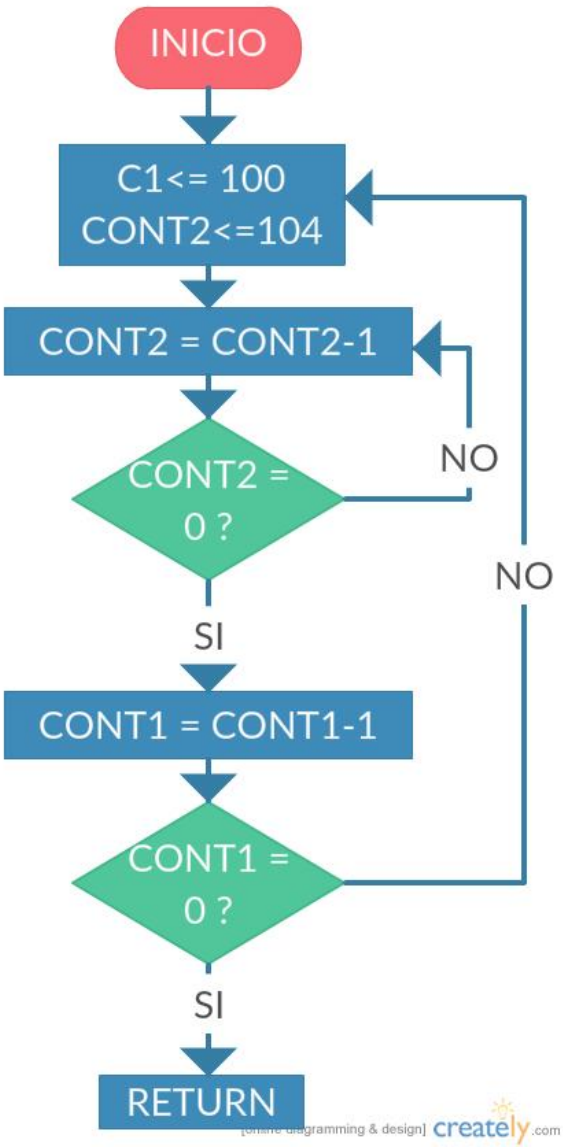
[7]. Designing Embedded Systems with PIC Microcontrollers Principles and applications. Tim Wilmshurst. 2007. Elsevier

VIII. ANEXOS

Los diagramas de flujo presentados se hicieron con ayuda de la herramienta online Creately, disponible en el link: https://creately.com/app/?tempID=hp21djew1#

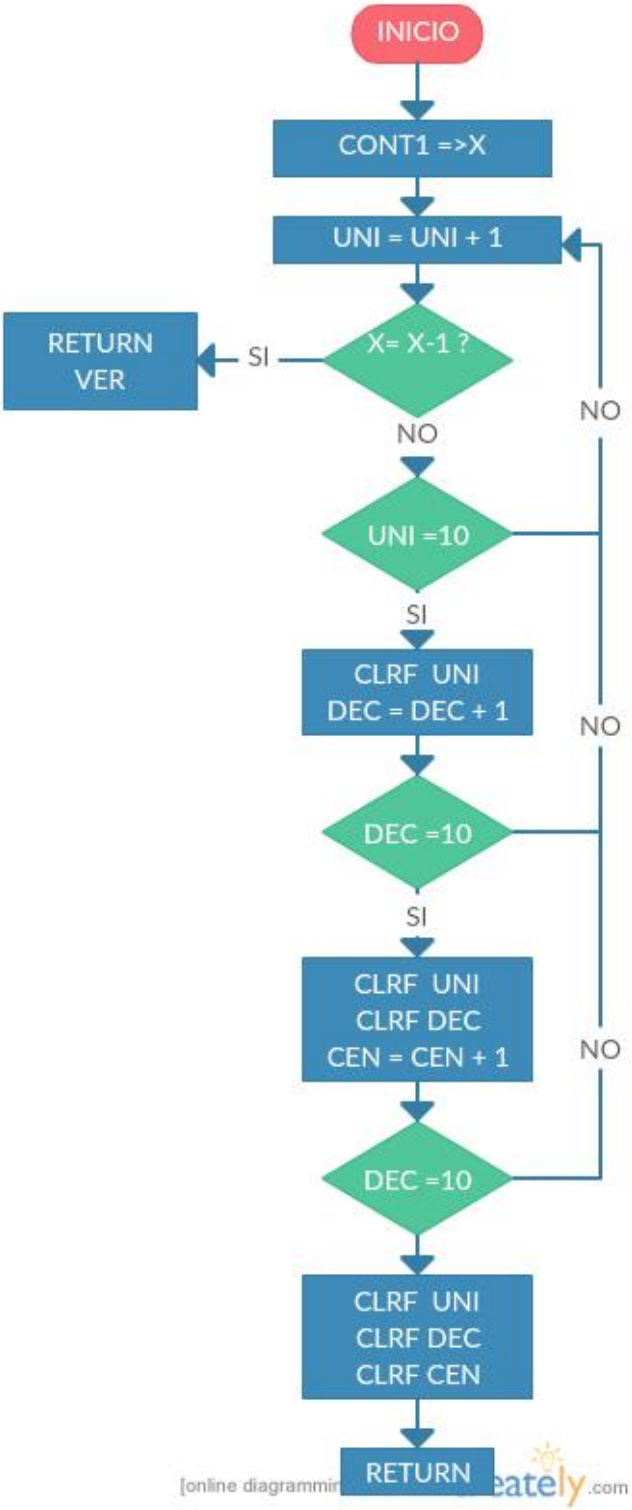


Anexo 1. Diagrama de flujo correspondiente al programa principal.

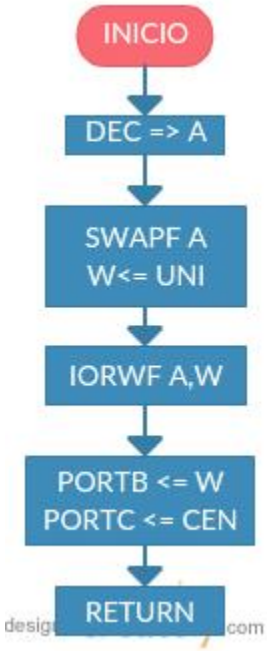


Anexo 2. Diagrama de flujo correspondiente a la subrutina del retardo.

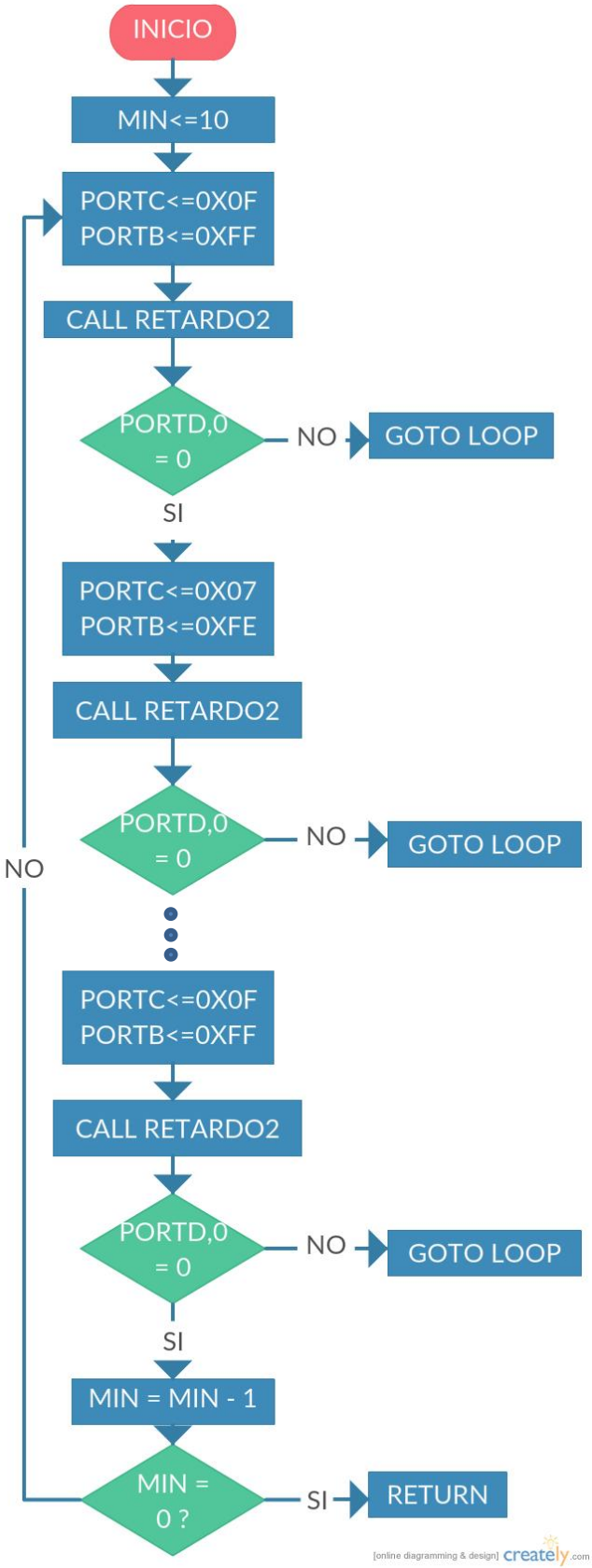
Con base en esta subrutina se hace lo mismo para el retardo de la secuencia, la cual tiene un valor de 54 en el CONT2, con exactamente la misma estructura.



Anexo 3. Diagrama de flujo correspondiente a la subrutina del conversor.



Anexo 4. Diagrama de flujo correspondiente a la subrutina ver.



Anexo 4. Diagrama de flujo correspondiente a la subrutina de la secuencia.

En este caso se abrevió el diagrama de flujo debido a la extensión de este. Se asignan 12 valores al puerto B y C, los cuales hacen una secuencia de 6 segundos, que se repetirá 10 veces, hasta que la variable MIN sea 0.