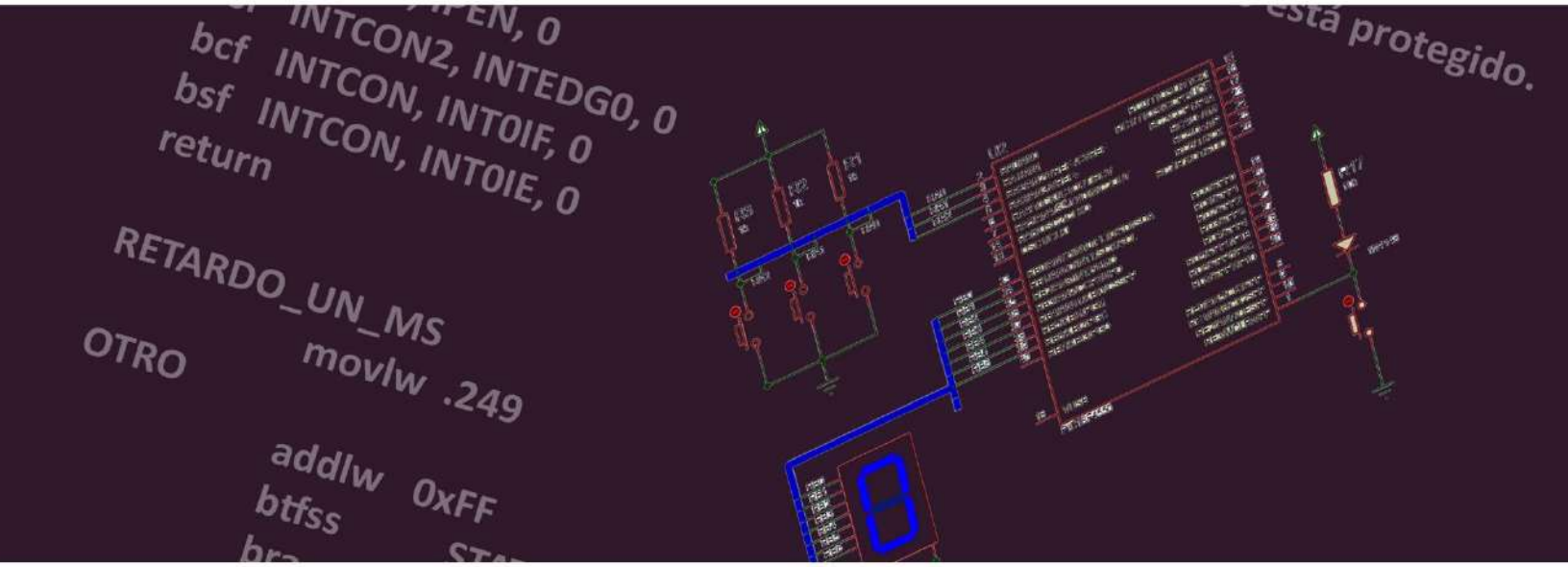
EXPERIMENTOS PRÁCTICOS

EN LENGUAJE ENSAMBLADOR CON EL

PIC18F4550



ITSON UNIVERSIDAD

Eduardo Romero Aguirre

Darcy Daniela Flores Nieblas

Experimentos prácticos en lenguaje ensamblador con el PIC18F4550

Eduardo Romero Aguirre
Darcy Daniela Flores Nieblas

ITSON UNIVERSIDAD

Instituto Tecnológico de Sonora
5 de febrero, No. 818 sur, colonia Centro
Ciudad Obregón, Sonora, México; 85000
www.itson.mx
Email: rectoria@itson.mx
Teléfono: (644) 410-90-00

Primera edición
Junio, 2025
ISBN para ebook: **978-607-609-266-8**

Gestión editorial
Marisol Cota Reyes
Oficina de publicaciones ITSON
marisol.cota@itson.edu.mx

Maquetación y diseño
Darcy Daniela Flores Nieblas
darcy.flores20554@potros.itson.edu.mx

Cubierta diseñada por: Darcy Daniela Flores Nieblas

La presente publicación ha sido dictaminada bajo un proceso doble ciego por pares académicos nacionales expertos en la línea temática de la obra.

Hecho en México

DIRECTORIO

Dr. Jesús Héctor Hernández López
Rector

Dr. Jaime Garatuza Payán
Vicerrectoría Académica

Dr. Rodolfo Valenzuela Reynaga
Vicerrectoría Administrativa

Dr. Ernesto Uriel Cantú Soto
Secretario de la Rectoría

Mtro. Mauricio López Acosta
Dirección Unidad Navojoa

Mtro. Humberto Aceves Gutiérrez
Dirección Unidad Guaymas

PRÓLOGO

La electrónica avanza constantemente en el ámbito digital, impulsando la evolución acelerada de los sistemas basados en microprocesadores y microcontroladores (MCUs). Su aplicación se extiende a computadoras, telefonía móvil, maquinaria industrial, electrodomésticos, dispositivos IoT y televisores inteligentes, entre otros.

Para mantener un aprendizaje acorde con estos cambios, el bloque de sistemas digitales presenta este libro de experimentos prácticos, diseñado para complementar los cursos teóricos de programación de microcontroladores en las carreras de ingeniería electrónica y la de mecatrónica.

El objetivo es proporcionar herramientas de hardware y software, junto con métodos, técnicas y procedimientos útiles empleados para el diseño de sistemas basados en MCUs, en especial aquellos centrados en el PIC18F4550. Se emplea el lenguaje ensamblador, cuya complejidad lo convierte en un reto de aprendizaje, pero también en una habilidad altamente demandada en el diseño digital profesional.

Este libro busca no solo proporcionar conocimientos técnicos fundamentales sobre el diseño con microcontroladores, sino también fortalecer la capacidad analítica y práctica del estudiante. Al sumergirse en experimentos concretos y casos de aplicación real, el lector podrá desarrollar una comprensión sólida del funcionamiento del PIC18F4550 y del lenguaje ensamblador, habilidades altamente valoradas en la industria. Con este enfoque estructurado, se espera que los futuros ingenieros estén mejor preparados para enfrentar los retos del diseño digital y contribuir con soluciones innovadoras en un campo en constante evolución.

Dr. Eduardo Romero Aguirre

C O N T E N I D O

TEMA 1

HERRAMIENTA DE DESARROLLO MPLAB X

INTRODUCCIÓN

MPLAB® X de Microchip es un programa de software expandible y altamente configurable que incorpora herramientas poderosas que permiten descubrir, configurar, desarrollar y depurar diseños digitales *embedded* para la mayoría de los microcontroladores y controladores digitales de señales de Microchip. MPLAB X® funciona perfectamente con el ecosistema de desarrollo de software y herramientas de software disponibles para MPLAB normal. Asimismo, cuenta con un visualizador de datos, amplia biblioteca de guías y recursos, visualizador de estados de los pines I/O y facilidad de uso. Estás características ayudan a minimizar los tiempos de desarrollo de proyectos de esta índole, que, en conjunto con los emuladores comerciales y dispositivos programadores, conforman un conjunto de herramientas muy completo para el trabajo y/o el diseño de sistemas *embedded* basados en este tipo microcontroladores.

Además, MPLAB® X ofrece una serie de características avanzadas diseñadas para optimizar la experiencia del desarrollador. Estas incluyen la capacidad de realizar simulaciones detalladas y análisis en tiempo real, lo que permite a los ingenieros identificar y resolver problemas de diseño antes de la implementación final. La integración con plataformas de control de versiones facilita la colaboración en equipo, mientras que las extensiones y *plugins* disponibles permiten personalizar el entorno de desarrollo según las necesidades específicas del proyecto.

El software también admite una amplia gama de periféricos y módulos, lo que amplía su versatilidad y utilidad en diversos campos de aplicación, desde la automoción y la industria hasta la domótica y la electrónica de consumo. El soporte técnico de Microchip y la comunidad de usuarios proporcionan un respaldo invaluable, ofreciendo soluciones y respuestas a problemas comunes y complejos.

En resumen, MPLAB® X no solo acelera el desarrollo y la depuración de proyectos *embedded*, sino que también ofrece un entorno robusto y adaptable que se adapta a las necesidades cambiantes de los desarrolladores y las demandas del mercado.

El siguiente experimento te permitirá conocer el entorno de desarrollo integrado (IDE) MPLAB® X de Microchip, así como:

- Identificar los pasos necesarios para crear un proyecto de programación en ensamblador para microntroladores PIC.

- Describir los procesos requeridos en MPLAB® X para editar, depurar y simular un proyecto codificado en lenguaje ensamblador para un MCU PIC.
- Configurar el software para el dispositivo programador de PIC's (compatible con el PIC18F4550) para la descarga a su memoria interna del código máquina (archivo HEX) generado en MPLAB® X.

A continuación, se presenta una lista de material y equipo que se necesitará para el correcto desarrollo del experimento:

Cantidad	Descripción
1	Microcontrolador PIC18F4550
1	Diodo IN4148
1	Push-button
8	Leds o una barra de leds
8	Resistencia de 330 Ω
1	Resistencia de 1 kΩ
1	Tableta experimental

- Dispositivo programador compatible con PIC18F4550.
- Fuente de alimentación de CD.
- Laboratorio equipado con computadoras que tengan instalado el MPLAB® X versión v5.0 o superior y software para dispositivo programador de PIC's.

Antes de comenzar se sugiere realizar las siguientes actividades:

- Leer previamente todo el documento del experimento 1.
- Descargar el manual de usuario del dispositivo programador de PIC's que vaya a usar para futuras consultas
- Investigar el mapa de memoria de datos donde muestra la ubicación de cada uno de los registros SFR (información disponible en el *datasheet* del PIC18F4550 o del archivo de encabezado con extensión INC).
- Implementar en un *protoboard* el circuito de la figura 1.18.

EXPERIMENTO 1

DESARROLLO

I. Procedimiento para crear un proyecto en MPLAB X

1. Para ingresar al programa MPLAB IDE, basta con hacer doble *click* en el icono instalado en el escritorio de la PC, o seleccionar ***Inicio> Programas> Microchip> MPLAB X IDE v5.30***. Lo anterior desplegará la presentación del software seguido de la pantalla principal (figura 1.1).

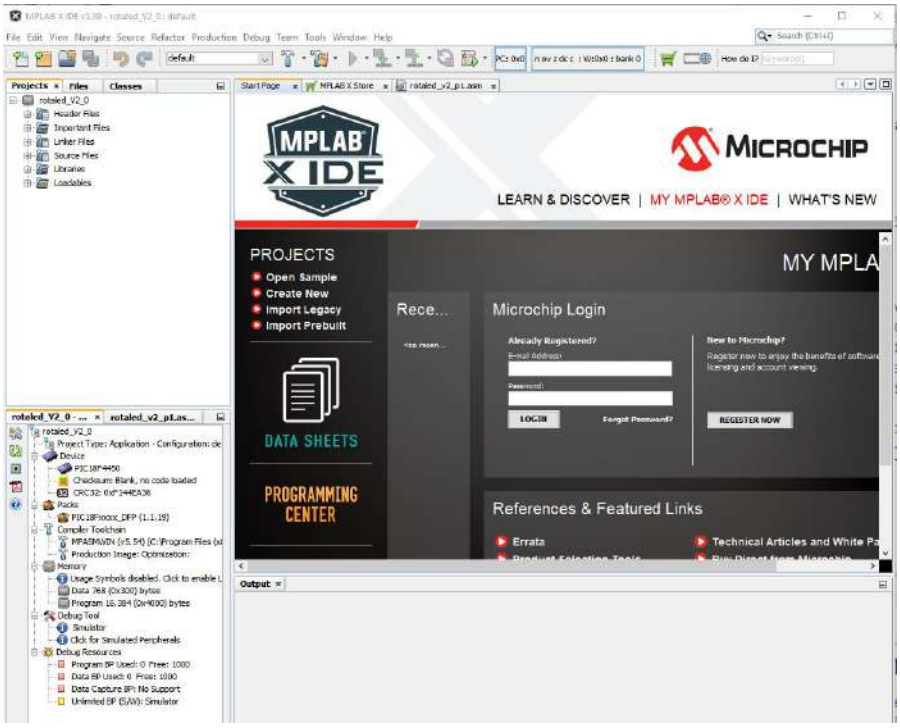


Figura 1.1: Pantalla principal de MPLAB X®.

2. MPLAB X® se basa en proyectos, por lo que es necesario crear uno para trabajar en su aplicación. Para poner en marcha un nuevo proyecto. En la página de inicio, en la pestaña ***Learn*&discover**, en la sección ***Projects***, elegir la opción ***Create new***. Lo anterior desplegará una ventana similar a la que se muestra en la figura 1.2.

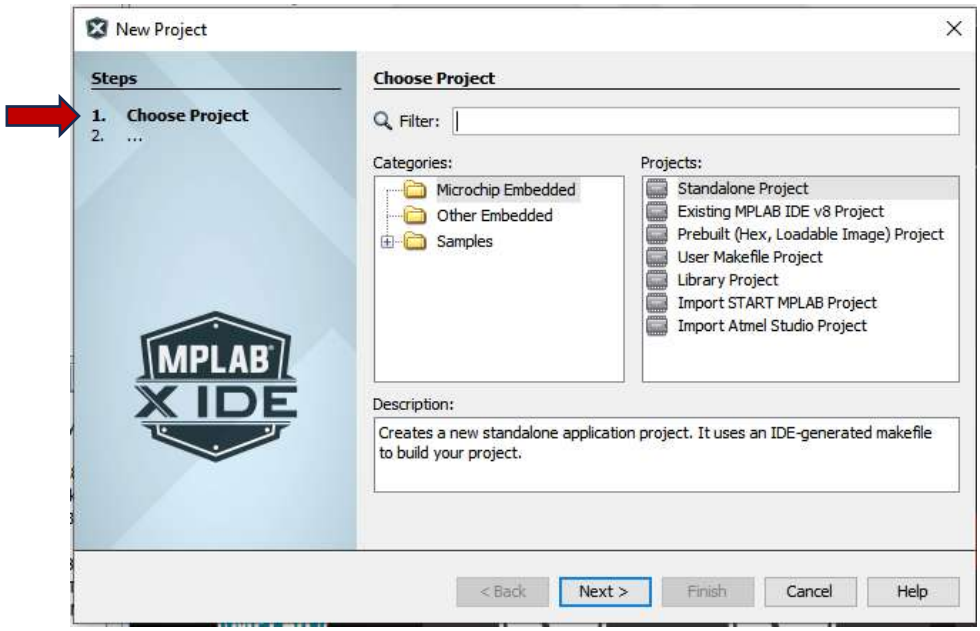


Figura 1.2: Ventana *New Project*

3. En la ventana ***New Project***, se debe elegir la categoría de proyecto, en nuestro caso Microchip *embedded*. Después, hay que escoger uno de los ocho tipos de proyectos. Seleccione el ***Standalone Project*** (Proyecto independiente) y presione ***Next***. Esto abrirá la ventana ***Select Device***, mostrada en la figura 1.3. Elija la familia y dispositivo que se muestran y presionar ***Next***.

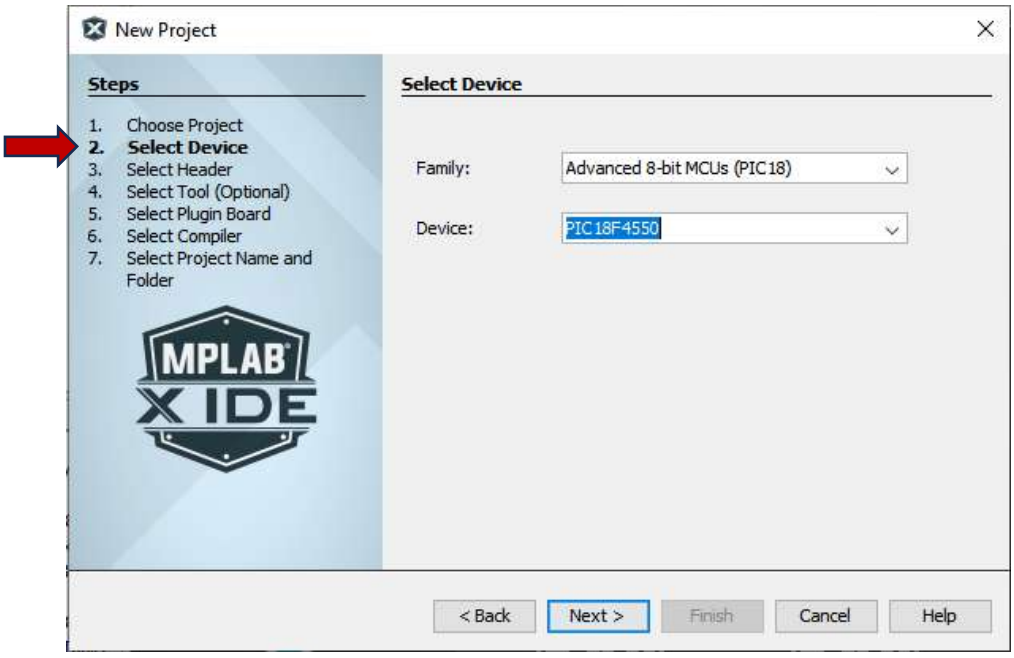


Figura 1.3: Ventana de la opción 2 *Select device.*

4. A continuación, se despliega la ventana ***Select header***, este paso es opcional y está relacionado a agregar algún archivo de cabecera para propósitos de depuración. No se hará uso de esta opción, por lo que simplemente se avanzará al siguiente menú presionando ***Next***.

5. La siguiente ventana que se despliega es la denominada ***Select tool***, la que también es opcional. Aquí es posible seleccionar alguna herramienta de hardware de Microchip o de terceros. Debido a que esto no es nuestro caso, se avanza al siguiente menú.

6. En la ventana ***Select compiler*** (figura 1.4), se selecciona la herramienta del lenguaje de programación a emplear, ya sea un compilador C o ensamblador. En esta práctica se usará la versión 5.54 del ensamblador MPASM.

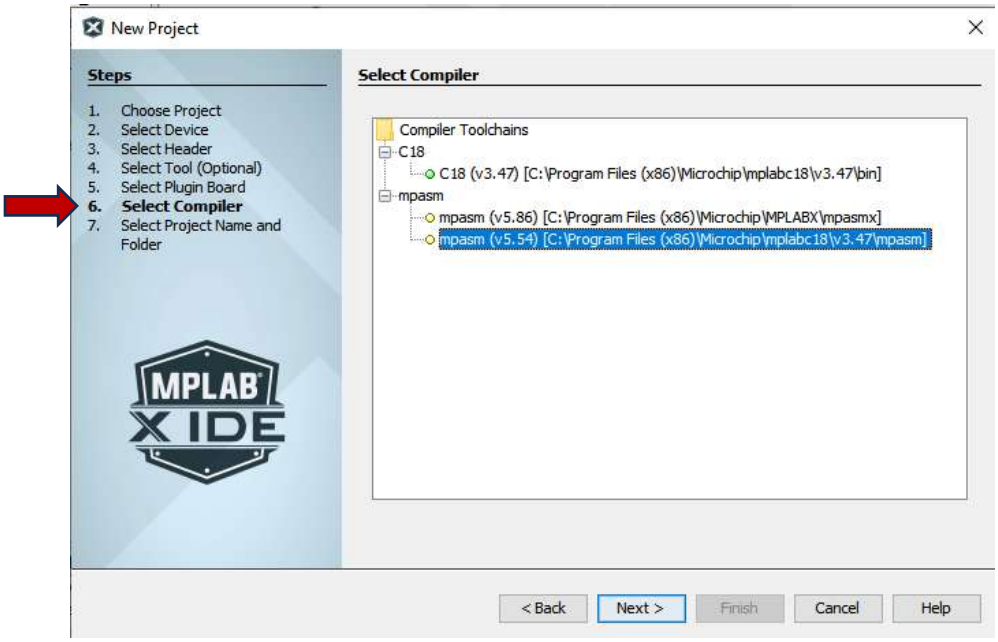


Figura 1.4: Ventana de la opción 6, *Select compiler*.

7. En la ventana ***Select Project Name and Folder***, será donde se le debe dar un nombre al proyecto (practica_1) y se deberá elegir una ubicación donde se va a almacenar. Se puede optar por la que usa la herramienta en forma predeterminada o alguna diferente. En esta práctica se usará la que se muestra en la figura 1.5. También se debe marcar la casilla ***Set as main project*** para indicar que el mismo va a ser su proyecto principal y la opción de ***encoding*** se debe quedar tal como lo propone MPLAB X®.

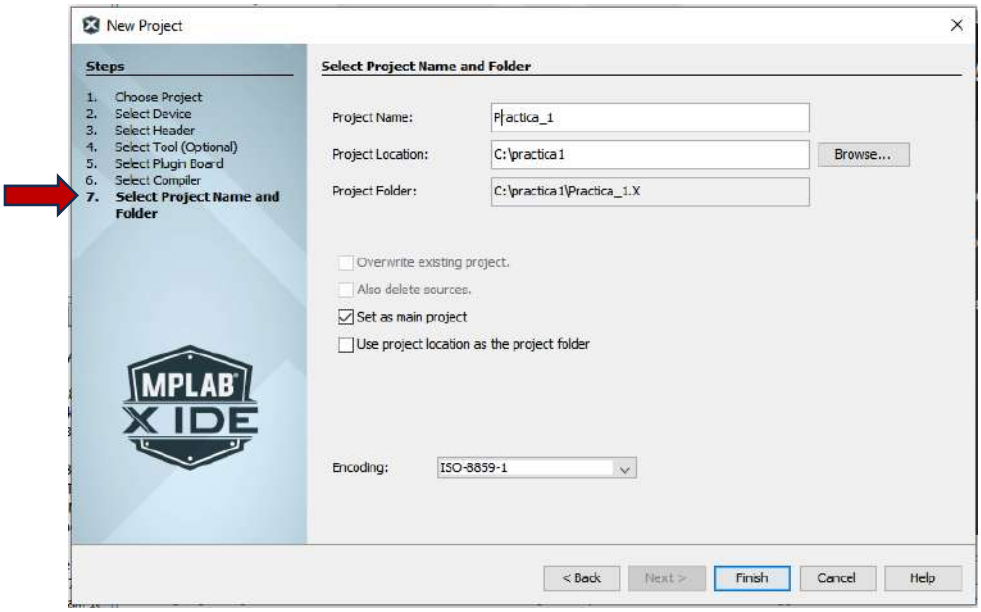


Figura 1.5: Ventana de la opción 7, *Select Project Name and Folder.*

8. Una vez concluida la creación del proyecto, el mismo quedará conformado como un grupo de archivos dentro de una estructura definida de carpetas similar a la mostrada en la figura 1.6. MPLAB X® interactúa dinámicamente con cada una de ellas para almacenar, recuperar y ordenar los archivos relacionados con el proyecto en cuestión.

Figura 1.6: Estructura de las carpetas de un proyecto vista desde el explorador de Windows.

También es posible visualizar tal estructura en la esquina superior izquierda de MPLAB X®, tal como se muestra en la figura 1.7.

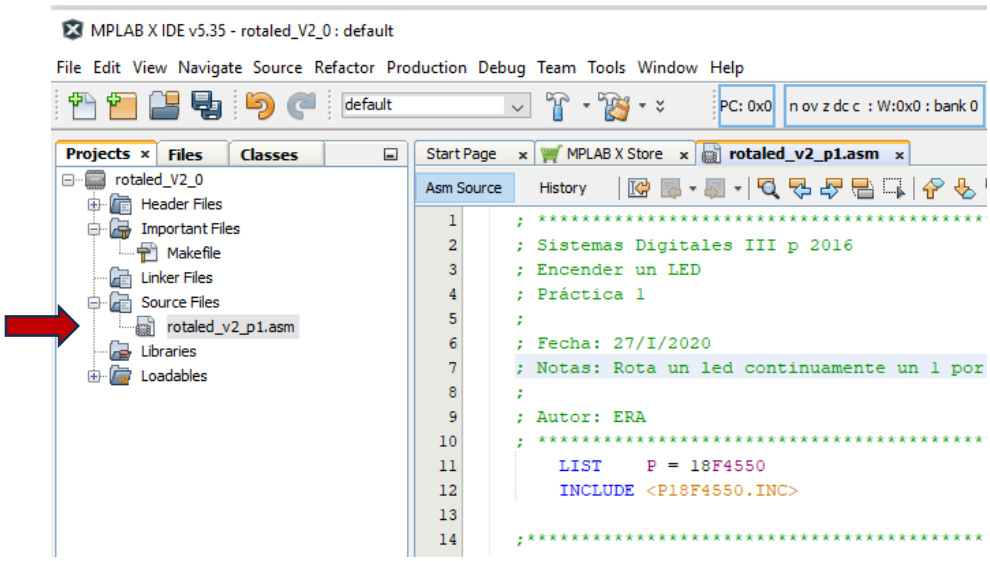


Figura 1.7: Estructura de las carpetas de un proyecto vista desde MPLAB X®.

9. Para añadir ficheros fuente ASM, basta con dar clic derecho en la ventana de proyecto en la carpeta ***Source files***. En caso de archivos en ensamblador, seleccionar ***Other*** y luego ***Assembler*** para el tipo de archivo con extensión ASM (figura 1.8). Para continuar presione ***Next***.

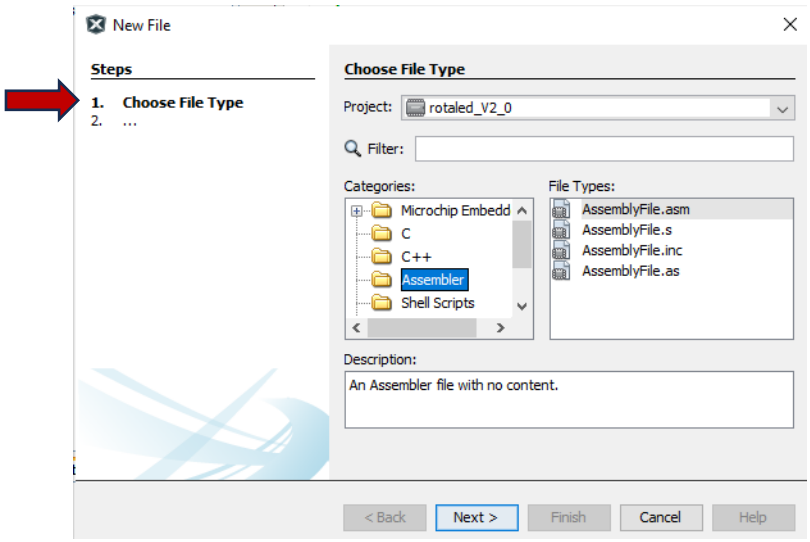


Figura 1.8: Añadir un archivo fuente ASM a un proyecto.

10. Se desplegará una ventana donde se introducirá el nombre y la ubicación del archivo fuente (figura 1.9). Para facilitar su identificación, es recomendable asignar un nombre que refleje lo que se pretende hacer en el programa.

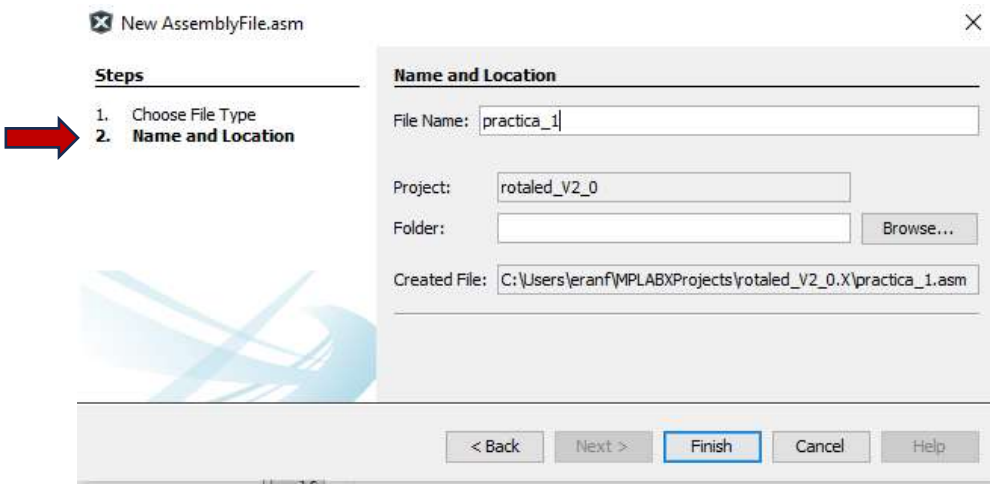


Figura 1.9: Asignando nombre y ubicación del archivo fuente ASM en un proyecto.

11. Hecho esto, presionar el botón de ***Finish*** y a continuación se abrirá la ventana del editor en ensamblador. A continuación, escriba el código del programa fuente en ensamblador que se muestra en la figura 1.10 y asígnele el nombre de **rotaled.asm**.

```
;*******************************************************************************
; Sistemas Digitales III p 2016
; Encender un LED
; Práctica 1
;
; Fecha: 27/I/2020
; Notas: Rota continuamente un bit por el puerto B
; Autor: ERA
;*******************************************************************************
    LIST    P = 18F4550
    INCLUDE <P18F4550.INC>

;*******************************************************************************
;               BITS DE CONFIGURACION
;*******************************************************************************
;********       Configuración del Oscilador     **********
    CONFIG  FOSC    = INTOSCIO_EC  ;Osc interno, RA6 pin libre

;********      Bits de configuración mas usados
**************************
    CONFIG  PWRT    = ON        ;PWRT habilitado
    CONFIG  BOR     = OFF       ;Brown out deshabilitado
    CONFIG  WDT     = OFF       ;Watchdog deshabilitado
    CONFIG  MCLRE   = ON        ;MCLR como entrada de reset
    CONFIG  PBADEN  = OFF       ;PB como entradas digitales
    CONFIG  LVP     = OFF       ;Prog bajo voltaje apagado
    CONFIG  DEBUG   = OFF
    CONFIG  XINST   = OFF
```

Figura 1.10: Código fuente en ensamblador para el circuito experimental con el PIC18F4550.

```
;*********      Bits de proteccion
************************************
    CONFIG  CP0     = OFF       ;los bloques del codigo de programa
    CONFIG  CP1     = OFF       ;no estan protegidos
    CONFIG  CPB     = OFF       ;Sector Boot no esta protegido

;********* Definición de variables
************************************
 ; Variables en la ACCESS RAM
    CBLOCK  0x00
    CANT_MS:2                   ;Variable para generar 65535 s
    ENDC

;*****************************************************************************
; Vector de Reset.
    ORG     0x0000
    goto    INICIO              ;Ve al inicio del programa principal

;*****************************************************************************
; Vector de interrupcion de baja prioridad y rutina
    ORG     0x0018
    goto    INICIO              ;Ve al inicio del programa principal

;*****************************************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
    movlw   B'01100010'
    movwf   OSCCON              ;Oscilador interno a 4 MHz
    return

;*****************************************************************************
; Subrutina que configura todo el puerto B
; como puerto de salida y deshabilita comparadores
CONF_PUERTOB
    clrf    LATB, 0             ;Limpia Lacths del puerto B
    movlw   0Fh
    movwf   ADCON1, 0           ;Todos los pines como I/O digitales
    clrf    TRISB, 0            ;Todos loe pines de puerto B como salidas
    movlw   0x04
    movwf   LATB, 0             ;Puerto B = 0000 0100b
    return

;*****************************************************************************
; Subrutina que rota a la derecha un bit por el puerto B
ROTADER
    rrncf   LATB, f, .0         ;Rota a la derecha el puerto B
    rcall   RETARDO_MEDIO_SEG   ;Introduce retardo de 500 ms
    return
```

Figura 1.10: Código fuente en ensamblador para el circuito experimental con el PIC18F4550 (cont.)

```
;*********************************************************************
; Subrutina que genera un retardo de 500 ms aprox
; haciendo uso de la rutina RETARDO_VAR_MS
RETARDO_MEDIO_SEG
    movlw   0x03                ;No => hacemos Cantms = 1000 1f4
    movwf   CANT_MS+1,.0
    movlw   0xE8
    movwf   CANT_MS,.0
    rcall   RETARDO_VAR_MS
    return

;*********************************************************************
; Ejecuta un retardo variable que depende de la cantidad de ms indicados
; mediante las variables CANT_MS y CANT_MS+1
RETARDO_VAR_MS
    rcall   RETARDO_UN_MS       ;realizamos un retardo de 1 ms
    decfsz  CANT_MS, F,.0       ;Cantms --, Cantms == 0
    bra     RETARDO_VAR_MS      ;no => vamos a
    movf    CANT_MS+1, W, .0
    btfsc   STATUS, Z           ;Cantms+1 == 0?
    return                      ;Si => retornamos Retardo_ms
    decf    CANT_MS+1, F, .0    ;No => decrementamos
    bra     RETARDO_VAR_MS      ;Continuamos con el ciclo

;*********************************************************************
; Subrutina (bloqueante) que ejecuta un retardo de 1 ms por software
; decrementado el registro WREG 249 veces en un ciclo, asumiendo que la
; frecuencia de reloj es de un 1 MHz
RETARDO_UN_MS
    movlw   .249
OTRO
    addlw   0xFF                ;Decrementa W
    btfss   STATUS,Z,.0         ;¿Es igual a 0?
    bra     OTRO                ;No => seguimos esperando
    return                      ;Si => ya transcurrió 1 ms

;*********************************************************************
;       PROGRAMA PRINCIPAL
INICIO
    call    CONF_BASE_TIEMPO
    call    CONF_PUERTOB
AGAIN
    call    ROTADER
    bra     AGAIN

    END
```

Figura 1.10: Código fuente en ensamblador para el circuito experimental con el PIC18F4550 (cont.)

II. Depurado y simulación de un programa en ASM con MPAB® X

12. Toda vez editado y almacenado el archivo fuente dentro del proyecto, será necesario depurarlo para verificar y corregir los errores que se cometieron al momento de la codificación. Ensamble el archivo rotaled.asm, contenido dentro del proyecto, con cualquiera de los siguientes iconos:

- **Build Project**: Compila/ensambla solos los archivos que cambiaron desde la última vez.
- **Clean and Build Project**: Compila/ensambla los archivos indiferentemente si han cambiado o no. Adicionalmente elimina archivos temporales.

Cualquiera de las dos opciones generará un archivo objeto que suele estar formado por un código intermedio o código objeto y otro más de optimización de código cuyo fin es obtener el código más eficiente.

13. MPLAB X deberá entonces desplegar si existió un error y entonces se debe proceder a su interpretación y corregirlo. Esto puede verificarse en la pestaña correspondiente en la ventana ***Output*** (figura 1.11). En caso de que se generen *warnings* y/o errores, haga doble clic donde se marquen (con rojo) en la ventana Output, corrija y repita el proceso hasta que se indique un ensamble exitoso (***build successful***).

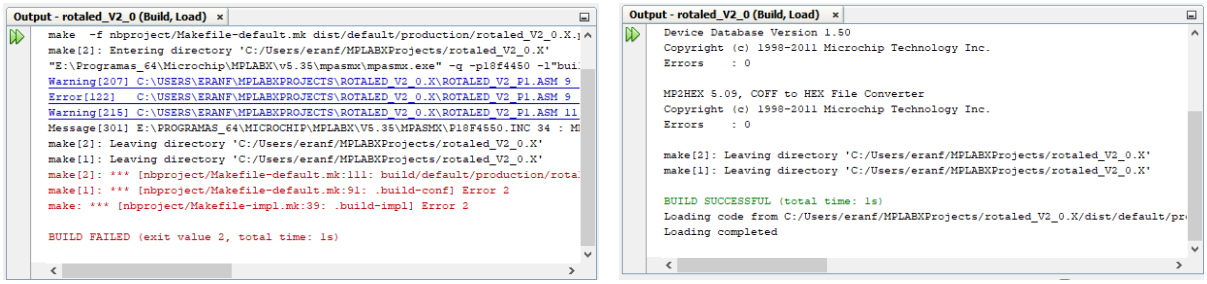


Figura 1.11: Ejemplos de ensambles en la Ventana **Output**: Izquierda: ensamble fallido con errores y *warnings*. Derecha: ensamble exitoso sin errores.

14. Para simular el código del programa, es necesario seleccionar la herramienta desde **File >Project Properties** (nombre del proyecto) o bien desde el campo **default**, se elige **customize** y deberá aparecer una ventana similar a la de la figura 1.12. En caso de no aparecer en color, desbloquear por medio del botón **unlock**. Proceda a configurar la ventana con las opciones que se muestran en la figura 1.12.

15. Ejecute la herramienta de depuración por medio del menú ***Debug >Debug Project***. El programa se compilará, programará de forma opcional el PIC si se tiene configurado un programador compatible con MPLAB® X y comenzará a simular el código.

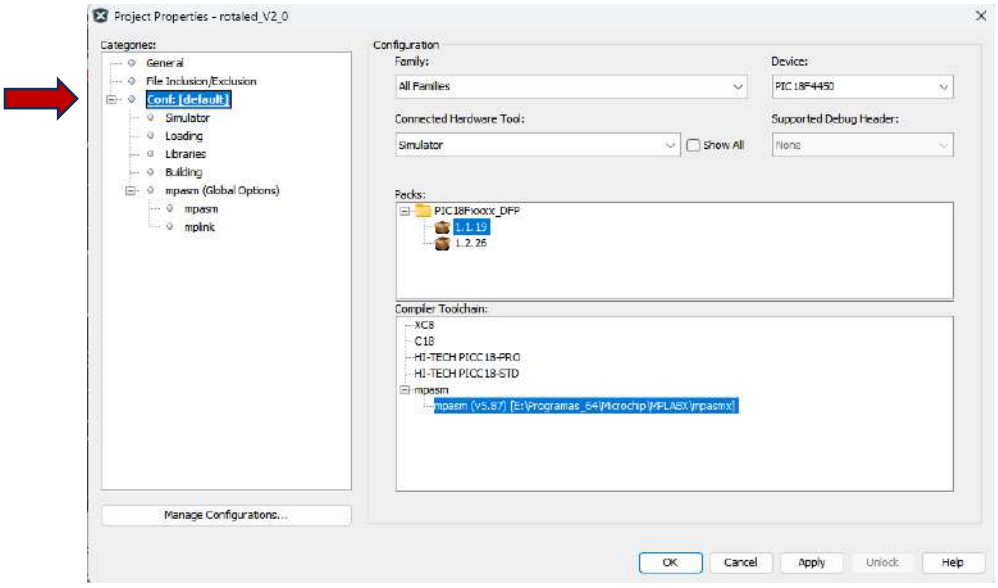


Figura 1.12: Ventana del menú *Debug* con la opción *Debug Project* seleccionada

16. Después de hecho lo anterior, se procede a simular un *reset* del microcontrolador por medio del menú ***Debug >Reset*** o usando el ícono . El puntero de color verde se posicionará en la primera instrucción de memoria, que por lo regular es un ***goto INICIO***, tal como se muestra en la figura 1.13.

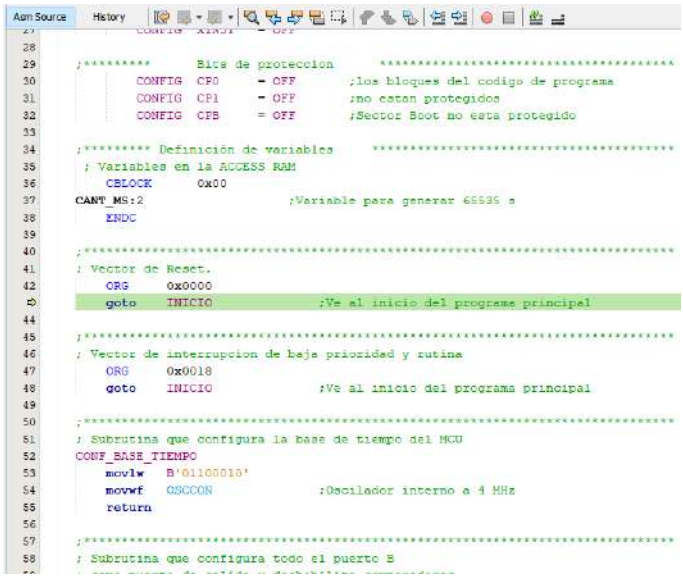


Figura 1.13: Ventana del código después de simular un *reset*

17. Luego de lo anterior, es posible simular el código fuente en ensamblador usando las opciones disponibles en la barra de herramientas de simulación (tabla 1.1).

Tabla 1.1: Resumen de los botones de la barra de herramientas de simulación

Menu Debug	Barra de herramientas	Tecla rápida
Finish Debugger Sesion		Shift + F5
Pause		Ctrl + Alt + 8
Reset		
Continue		F5
Step Over		F8
Step Into		F7
Step Out		Ctrl + F7
Run to Cursor		F4
Set PC at Cursor		
Focus Cursor at PC		

18. Para visualizar los cambios en cada una de las zonas de memoria interna del PIC durante la simulación, se debe seleccionar el menú ***Windows >Target Memory Views*** (figura 1.14), para elegir alguno de los que se enumeran a continuación.

- ***File registers***: Muestra los cambios de todos los registros (GPR y SFR) de la memoria de datos usados en el programa durante su ejecución. Una forma útil de visualizarlos es eligiendo la opción ***Symbol*** del menú desplegable ***Format***.

- ***SFR's***: Sirve para visualizar los registros especiales del microcontrolador. Es posible mostrador en orden de su ubicación en la memoria con la opción ***Individual*** del menú desplegable ***Format***, o agrupados de acuerdo con el periférico interno relacionado (opción ***Peripherals***).

- ***Configuration bits***: Los parámetros de los bits de configuración como el tipo de oscilador, protección de memoria, perro guardián, etc, se pueden seleccionar con esta opción e insertarse en el código.

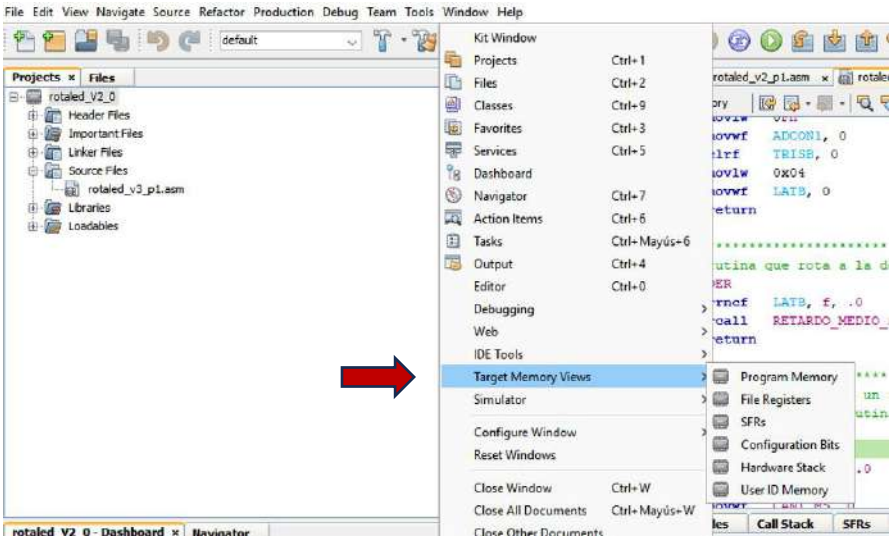


Figura 1.14: Ventana del menú *Windows* con la opción *Target Memory Views* seleccionada.

19. Durante la simulación, con la secuencia ***Debug >New Watch***, se puede ver el contenido (incluso a nivel bit) de los registros asociados tanto a los registros GPR y SFR y para corroborar su correcto funcionamiento (figura 1.15).

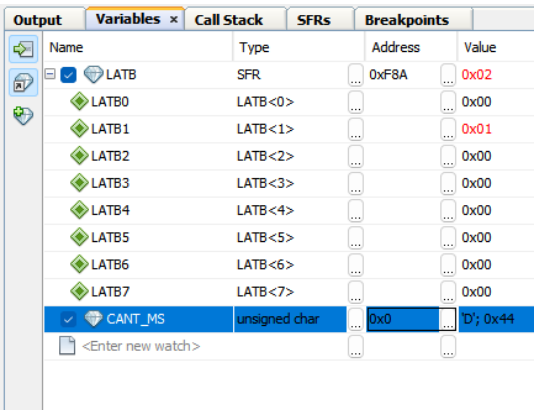


Figura 1.15: Ventana del menú *Debug* con la opción *New Watch* seleccionada.

20. Cuando se depura un programa a veces se requiere detener su ejecución en alguna instrucción específica. Esto puede hacerse insertando puntos de ruptura (*breakpoints*). Estos ayudan a observar los valores de los registros y variables en dicho instante. La inserción de un punto de ruptura se logra posicionando el puntero en el número de línea de una instrucción, dando *click* derecho de *mouse* y eligiendo la opción **(MPLAB X) *Breakpoint*** (figura 1.16).

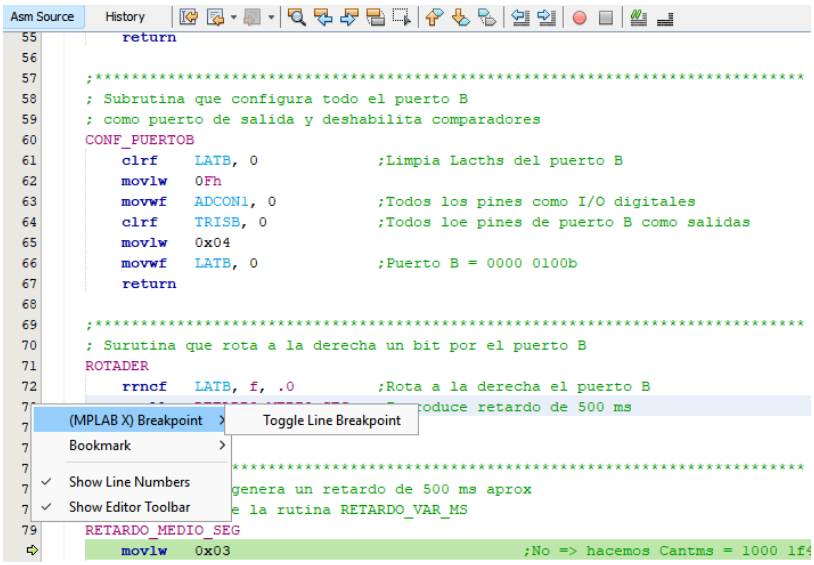


Figura 1.16: Ventana del menú contextual *(MPLAB X) Breakpoint* para insertar punto de ruptura

21. Otra forma de insertar un punto de ruptura es dando *click* izquierdo del apuntador rápidamente sobre el número de instrucción correspondiente. Se notará entonces que se agrega una marca en la parte lateral izquierda de la instrucción, tal como lo sugiere la figura 1.17. Oprima nuevamente el mismo botón si se quiere eliminar tal *breakpoint*.

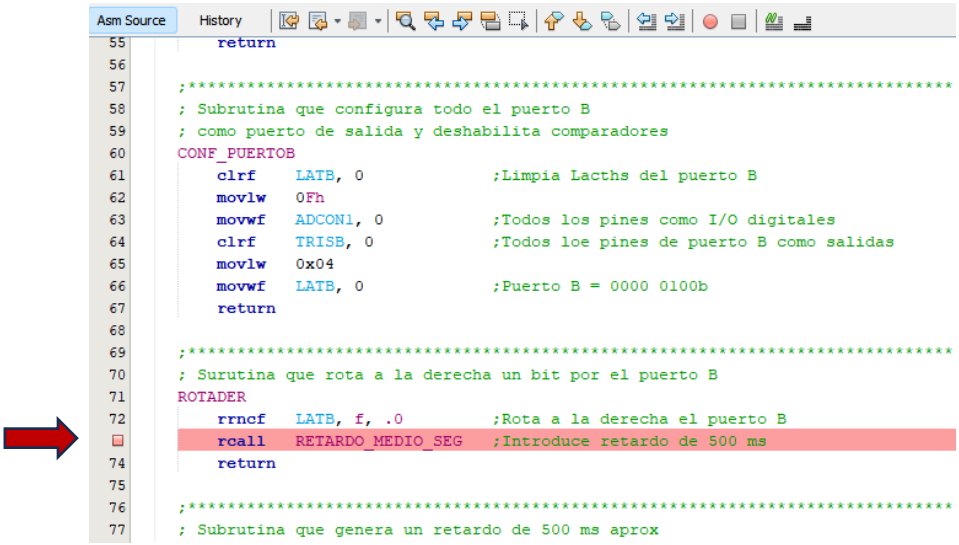


Figura 1.17: Listado del programa con el punto de ruptura insertado.

22. Empleando lo explicado en pasos anteriores, proceda a simular el código ASM ensamblado. Analice y visualice los registros, variables y pines de E/S involucrados en la ejecución del programa para validar que su funcionamiento sea el deseado.

23. Toda vez simulado y depurado el programa, programe el archivo HEX al microcontrolador con la ayuda el dispositivo programador que se tenga disponible.

24. Monte el PIC18F4550 en el circuito mostrado en la figura 1.18 y compruebe su funcionamiento.

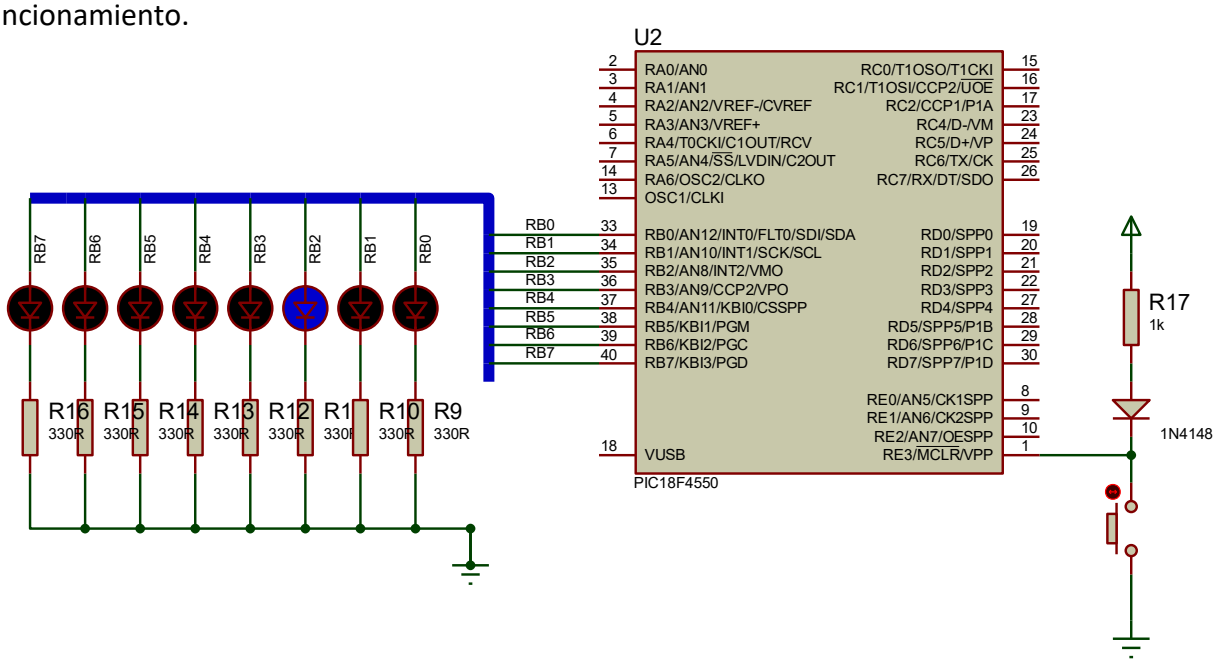


Figura 1.18. Circuito experimental para el código fuente de la figura 1.10

¿Explique el funcionamiento del circuito?

_____

_____

_____

25. Experimente con diferentes valores iniciales en el puerto B.

ACTIVIDADES COMPLEMENTARIAS

1. Investigue los programadores compatibles con el PIC18F4550 más empleados, Documente las diferencias que existen en el proceso de descarga del archivo HEX. Compare ventajas y desventajas. Concluya.
2. Investigue otras formas para “inicializar” los bits de configuración a los MCU PIC, analice ventajas y desventajas de cada opción. ¿Cuál recomendaría para el desarrollo de un proyecto? Justifique su respuesta.
3. Investigue si los programadores universales disponibles en el laboratorio de electrónica de su institución son compatibles con el MCU PIC18F4550.

TEMA 2

SIMULACIÓN Y DEPURACIÓN DE SISTEMAS BASADOS EN MCU'S CON PROTEUS

INTRODUCCIÓN

Proteus Design Suite® es un entorno de desarrollo CAD de la empresa Labcenter Electronics Ltd, el cual integra diversas herramientas de software para las tareas más comunes en el desarrollo de proyectos electrónicos tales como: captura de esquemáticos, fabricación de circuitos impresos y simulación basada en PSPICE. Las herramientas que conforman a Proteus son los siguientes:

- **ISIS** ("**I**ntelligent **S**chematic **I**nput **S**ystem"). El cual es módulo de captura de diagramas esquemáticos.
- **VSM** ("**V**irtual **S**ystem **M**odelling"). El cuál es el módulo de simulación, incluyendo PROSPICE.
- **ARES** ("**A**dvanced **R**outing and **E**diting **S**oftware"). El cuál es el módulo para la realización de circuitos impresos o **PCB**.

El programa que se explorará a fondo a través del desarrollo de esta práctica es el programa de captura de esquemáticos ISIS, el cual nos permite representar en forma gráfica un circuito que posteriormente podrá ser simulado. Algunas de las características más relevantes que posee ISIS son:

- Librerías de componentes.
- Conexionado automático entre dos puntos del esquema.
- Netlist compatible con la mayoría de los programas para la realización del PCB
- Enumeración automática de componentes, etc.

A través del siguiente experimento se pretende lograr los siguientes objetivos:

- Conocer los menús y opciones de la interfaz de usuario del software de desarrollo PROTEUS. y comprender los pasos para crear un proyecto de diseño con MCU's PIC.
- Identificar los pasos en PROTEUS para crear un proyecto en diseño esquemático con MCU's PIC.
- Descargar los archivos con código máquina HEX y COF al MCU empleado en el proyecto.
- Configurar las opciones de Proteus y MPLAB ® para visualizar la simulación/depuración interactiva en línea del código del MCU.

Para realizar el experimento 2 sólo se necesita contar con computadora que tenga instalado los programas PROTEUS versión 7.9 o superior y MPLAB® X v5.0 o superior.

Se aconseja realizar las siguientes actividades para lograr un desarrollo exitoso:

- Leer previamente todo el documento del experimento 2.
- Tener disponibles los archivos: fuente (ASM) y código máquina (HEX y COF) generados en el experimento 1.
- Contar con el documento del experimento 1.

EXPERIMENTO 2

DESARROLLO

I. Procedimiento para capturar un esquemático en ISIS de PROTEUS

1. Para ingresar al programa ISIS de PROTEUS, basta con hacer doble clic en el icono instalado en el escritorio de la PC, o seleccionar ***Inicio*** > ***Programas> Proteus Professional> ISIS Professional***. Lo anterior desplegará la presentación del software seguido de la pantalla principal (figura 2.1). Cabe mencionar que esto es válido sólo para la versión 7.9 o superior.

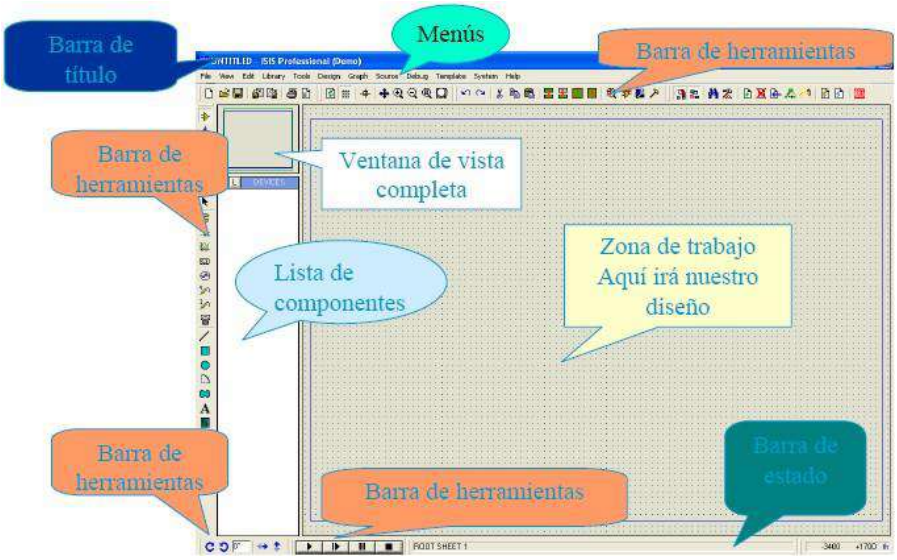


Figura 2.1: Pantalla principal de ISIS Professional.

La descripción de la función de cada uno de los componentes del software mostrados en la figura anterior se describe a continuación.

Barra de título: Situada en la parte superior de la pantalla, en ella se muestra el icono del programa, el nombre del fichero abierto (Apuntes), la leyenda ISIS Professional, y en ocasiones mensajes de que el programa ha entrado en un modo particular de funcionamiento (por ejemplo, ***Animating*** cuando se simula).

Barra de menús: Permite el acceso a la mayor parte de las opciones del programa, sin embargo, algunas opciones sólo están disponibles en los iconos de las barras de herramientas.

Barras de herramientas: Son varias y presentan las opciones para manejar los elementos del esquemático, tales como colocación de dispositivos, manejo de librerías, visualización, rotación de componentes, operaciones sobre bloques de dispositivos; así como las opciones de animación del diseño, entre otras.

Zona de trabajo: Es donde se colocará el diseño a realizar para posteriormente simularlo.

Ventana de vista completa/Zoom/Mapa del diseño: Esta ventana nos muestra una visión global del diseño, y mediante el puntero podemos seleccionar que zona del diseño estará visible en la ventana de edición, si no fuese posible visualizar todo sobre dicha ventana. La zona visible se encuentra encuadrada dentro de dicha ventana, mediante un recuadro verde.

Lista de componentes: En esta ventana aparecerán todos los componentes, terminales, pines, generadores, etc. que se quieran introducir en el diagrama esquemático, esta ventana dispone de 2 botones 

, los cuales nos permiten acceder a las librerías de componentes incluidas en ISIS.

Barra de estado: Situada en la parte inferior de la pantalla, en ella se muestran mensajes informativos acerca de las opciones del menú, de los componentes de las simulaciones a la derecha se indican las coordenadas de la posición del cursor. Las unidades son en milésimas de pulgada.

2. El proceso de captura de esquemas de circuitos electrónicos en ISIS consiste en realizar las siguientes tareas:

- Elegir en las librerías de componentes todos aquellos elementos que se utilizan en el circuito a realizar.
- Situar espacialmente los componentes que forman el circuito en la hoja de trabajo.
- Conectar los terminales de los componentes entre sí.
- Editar las propiedades de los componentes utilizados: valores nominales encapsulados etc. Para la colocación de dispositivos es necesario antes que nada establecer el modo componente, a través del icono  en la barra de herramientas localizada en el lado izquierdo de la pantalla.

Realizar la captura del diagrama esquemático mostrado en la figura 2.2. En este circuito se descargará un programa que desplace continuamente un bit por medio del puerto B.

Para seleccionar un dispositivo basta con presionar la tecla ***P***, o en la ventana de dispositivos el icono P. Lo cual nos mostrará la ventana ***Pick Devices***, como se observa en la figura 2.3.

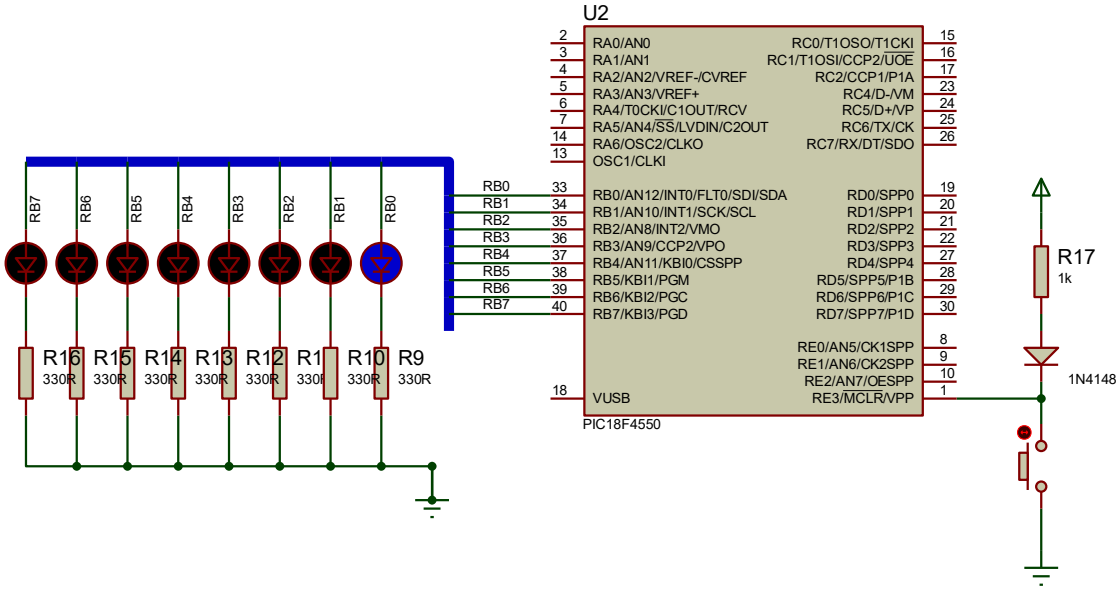


Figura 2.2: Circuito experimental para el programa de desplazamiento de bit por el puerto B.

Figura 2.3: Pantalla Pick Devices en ISIS.

Para seleccionar los componentes se deberá especificar primeramente la categoría (***Category***), seguido por la subcategoría (***Sub-category***) y finalmente el fabricante (***Manufacturer***). De manera alterna se puede teclear el número de parte del componente que deseamos agregar a la

lista de dispositivos en el campo ***Keywords***. Por ejemplo, para la selección de microcontrolador PIC18F4550, se seleccionan las opciones ***Microprocessors ICs > PIC18 Family > Microchip***, como se observa en la figura 2.4.

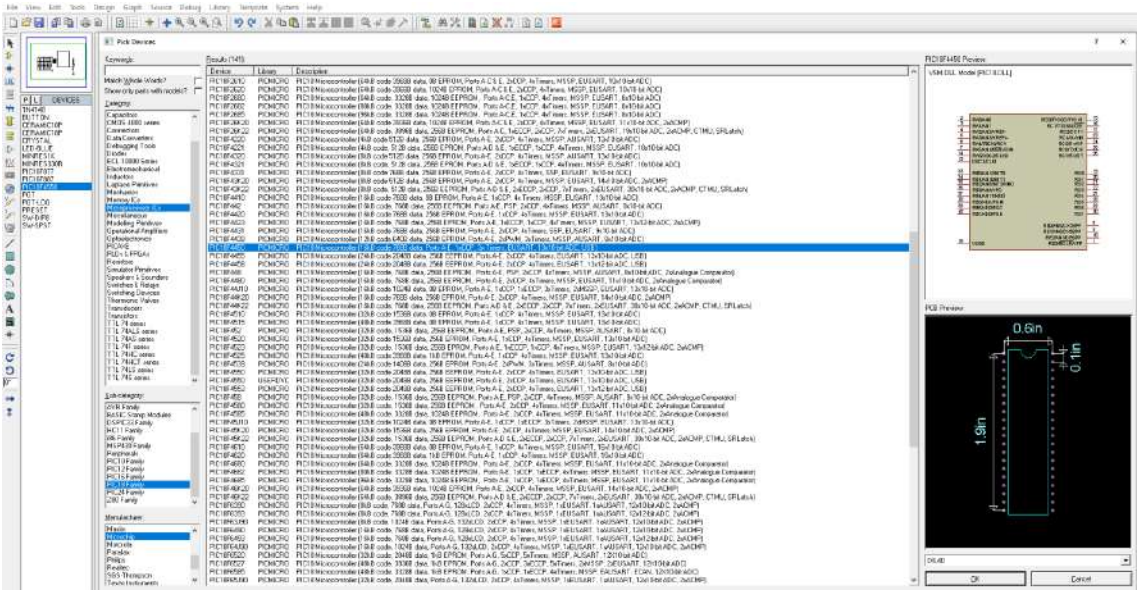


Figura 2.4. Pantalla *Pick Devices* en ISIS, selección del MCU PIC18F4550.

Al hacer doble clic en el dispositivo este se agrega a la lista; se pueden agregar cada uno de los componentes necesarios para el esquemático a la lista de dispositivos para posteriormente posicionarlos en el diagrama esquemático en la zona de diseño.

Agregue los dispositivos necesarios para el diagrama esquemático de la figura 2.2 a la lista de componentes. Utilice el campo de búsqueda *Keywords*, o navegue a través de las librerías para realizar la búsqueda. Las opciones para las partes del diagrama esquemático de la figura 2.2, se muestran en la siguiente tabla:

Tabla 2.1. Ubicación de componentes en las librerías de PROTEUS
para el diagrama de la figura 2.3.

Componente	***Category***	***Sub-category***	***Manufacturer***
MCU PIC18F4550	Microprocessor ICs	PIC18Family	Microchip
Resistencias	Resistors	Generic	(All manufacturers)
Diodo 1N4148	Diodes	(All Sub-categories)	(All manufacturers)
LEDS	Optoelectronics	Bargraph Displays	(All manufacturers)
Interruptor 1P1T MOM	Switches & Relays	Switches	(All manufacturers)

Una vez agregados los elementos a lista de dispositivos, se procede a colocarlos en la zona de diseño, para esto es necesario seleccionar de la lista creada anteriormente, el dispositivo. En este instante en la ventana de “vista completa” aparecerá el símbolo del dispositivo, al hacer clic sobre el área de trabajo se mostrará sólo el borde del mismo en color rosa. Cabe señalar que es posible rotar el símbolo del dispositivo con las teclas de “mas” (+) y “menos” (-). Una vez que se haya elegido la ubicación del símbolo, dar clic para colocarlo en el área de trabajo. Al repetir la tarea con el mismo dispositivo, PROTEUS autoenumerará los elementos. Para eliminar un componente basta con dar doble clic derecho sobre el dispositivo o utilizar la tecla ***supr*** del teclado.

Las fuentes de alimentación y tierra se encuentran en el modo de Terminales (***Terminals Mode***) de la barra de herramientas en el icono  . De clic en este icono y coloque las terminales de alimentación y tierra en el circuito, seleccionando de la lista de componentes ***POWER*** y ***GROUND*** respectivamente.

3. Posicione todos los elementos sobre el área de trabajo para que se muestren como en la figura 2.5.

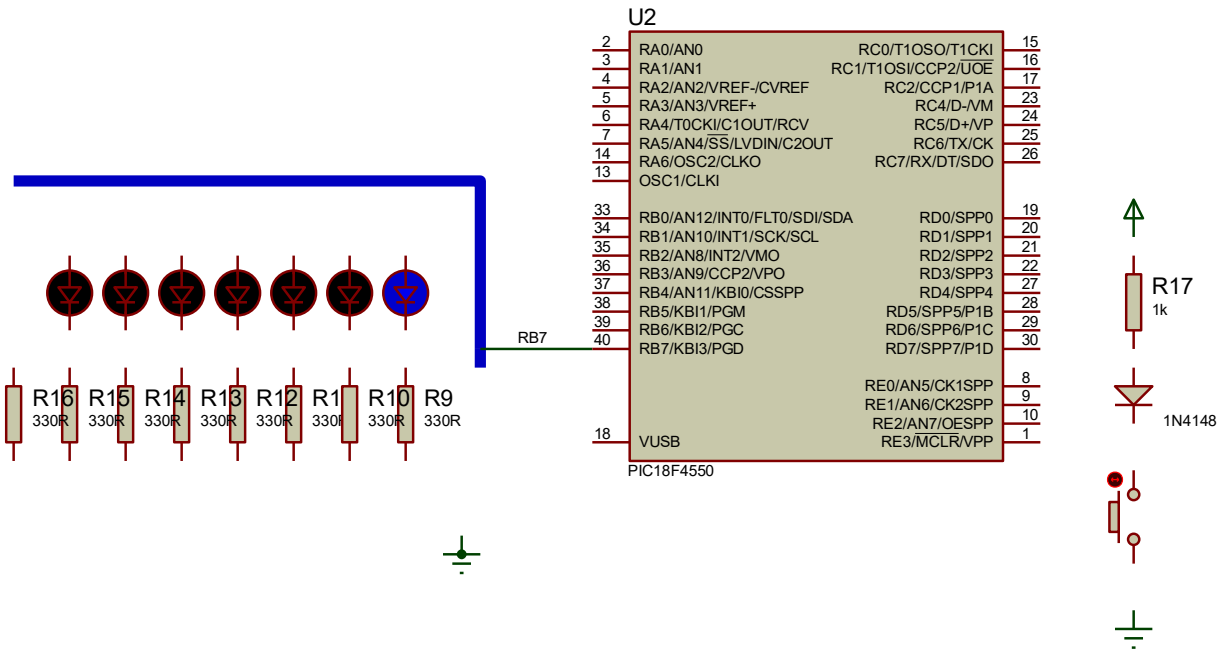


Figura 2.5: Diagrama esquemático sin conexiones entre dispositivos.

Para mover componentes basta con dar clic sobre los mismos, estos se tornarán de color rojo. Posteriormente posicione el cursor sobre el componente, el cual tomará la forma de una pequeña mano, de clic izquierdo y sin soltarlo arrastre la pieza al lugar de su preferencia, mientras

mueva el componente este tendrá un color rosa. Esta tarea también se puede realizar empleando la opción ***Block Move*** con el icono de la barra de herramientas, después de que el elemento se torna de color rojo al dar clic sobre él.

4. Una vez colocados los componentes es necesario hacer el conexionado de los mismos, para esto solamente posicione el cursor en el extremo del componente que desea conectar y de clic; en este momento el cursor empezará a dibujar una línea al mover el puntero, dirija el cursor hacia el extremo del otro dispositivo que desea conectar, cuando el pin del componente muestre un recuadro de un clic, si la conexión fue realizada correctamente ésta se dibujará entre ambas terminales; de no ser así repita el proceso.

Realice la conexión de todos los dispositivos hasta que el diagrama esquemático luzca como el mostrado en la figura 2.2. En caso de tener problemas con la visualización del circuito, auxíliese de los controles de ***zoom*** ubicados en la barra de herramientas, representados por los iconos: y con funciones similares a otros programas de diseño.

Una vez realizadas todas las conexiones entre los componentes del circuito se procederá a la edición de las propiedades de los componentes. Previo a esto guarde el archivo en el disco duro como si se tratara de cualquier otro tipo de archivo, de preferencia cree una carpeta en la raíz del disco duro y almacénelo con el nombre de **PRAC2**. Proteus agregará la extensión de forma automática al archivo.

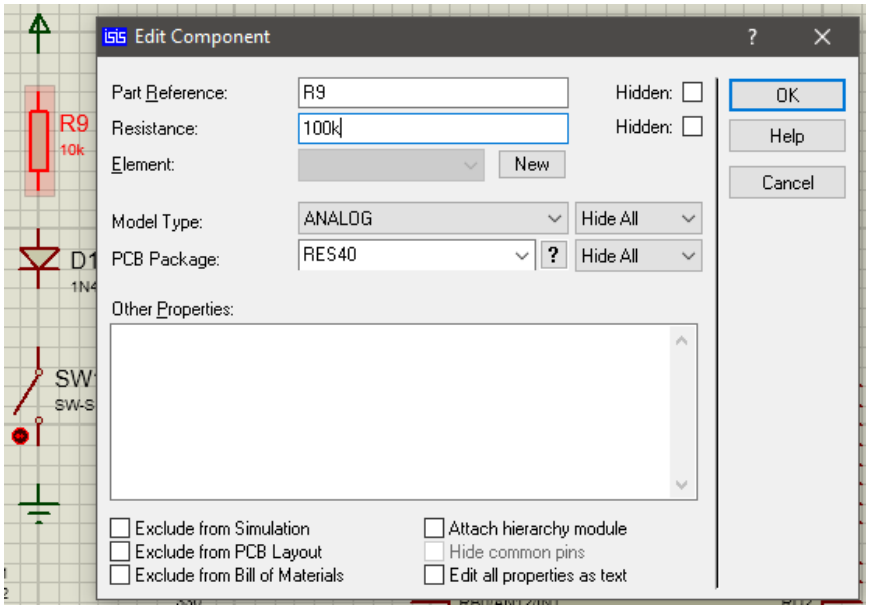


Figura 2.6: Cuadro de dialogo para edición de las propiedades de un componente.

La edición de las propiedades de los elementos del diagrama esquemático nos permite asignarles características específicas tales como el valor del componente, nombre especifico, así como también nos permite hacer visibles o no los atributos que muestran los mismos.

Para editar las propiedades de un componente, posicione el puntero del mouse sobre la etiqueta que desee modificar y de doble clic, con esto aparecerá una ventana en la cual podrá introducir el nuevo valor o nombre, como se muestra en la figura 2.6.

En el ejemplo de la figura anterior se cambió el valor de la resistencia **R1** de 10 kΩ a 100 kΩ, para establecer los cambios basta con dar clic en el botón **OK**. Adicionalmente las etiquetas de un componente pueden ocultarse, esto es posible haciendo doble clic sobre el componente y marcando la casilla ***Hidden***. En misma ventana puede editar los valores y nombres del elemento.

Vale la pena mencionar que al hacer edición en las propiedades de un componente se debe prestar especial cuidado en el campo del identificador asignado por PROTEUS, tal como R1, R2, C1, C2, etc., ya que en el caso de exportar el diagrama esquemático a un programa de fabricación de circuitos impresos se tendrían componentes duplicados, lo que provocaría problemas en la interconexión de las pistas y asignación de redes.

5. Edite todos los componentes del esquemático para que tengan los valores mostrados en la figura 2.2. Utilice la herramienta para generar el bus de conexión (línea azul) y edite la etiqueta en cada conexión individual de líneas para indicar su interconexión.

II. Simulación de circuitos basados en microcontroladores en Proteus

Dentro del programa de diseño PROTEUS, el programa VSM permite la simulación y animación de circuitos tanto analógicos como digitales. Para el desarrollo de esta práctica se cubren los aspectos relevantes relacionados con la simulación digital, haciendo especial énfasis en el microcontrolador. Una de las ventajas de contar con un simulador gráfico interactivo es que nos permite la visualización de los estados y respuesta del sistema, además en comparación con un simulador más simple como lo es el simulador de MPLAB X IDE®. Sin embargo, no hay necesidad de interpretar valores numéricos abstractos para saber el resultado y nos permite simular entradas de maneras similares al sistema final.

Aunque se utilice un simulador gráfico interactivo desarrollado por una empresa distinta al fabricante original de los microcontroladores —en este caso, Microchip— la precisión de los resultados, la respuesta del MCU ante eventos y la representación del comportamiento de los

periféricos dependen directamente del modelo implementado por el desarrollador del simulador, como lo es PROTEUS. Por ello, es fundamental tener precaución al interpretar resultados que se salgan de los parámetros esperados o cuando el comportamiento del sistema difiera significativamente del previsto, siempre posterior a una revisión minuciosa de las conexiones y de los parámetros configurados en cada componente del circuito.

Una vez realizado el diagrama esquemático del circuito con MCU con todas sus conexiones correspondientes y sus valores e identificadores editados, se procede a la simulación y verificación a través de la misma, del funcionamiento del código cargado a la memoria de programa del MCU.

6. Para "descargar" el archivo con extensión HEX al MCU, en el diagrama esquemático haga doble clic sobre el MCU, al hacer esto se le mostrará una ventana similar a la de la figura 2.7.

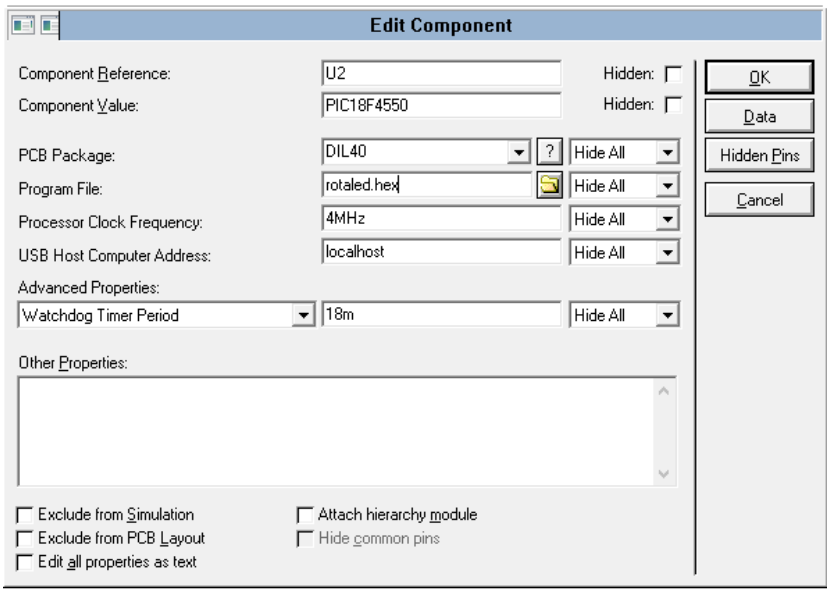


Figura 2.7: Cuadro de dialogo para la configuración de las propiedades del MCU PIC18F4550.

En el campo ***Program File*** de clic al icono  , el cual abrirá una ventana del explorador de Windows, a través de ésta busque la ruta del archivo solicitado en las actividades previas, correspondiente al archivo ***rotaled.hex*** de la práctica anterior. En la misma ventana determinaremos la frecuencia de trabajo del MCU, en el campo ***Processor Clock Frequency*** escriba **4MHz**, correspondiente a la frecuencia de trabajo del oscilador interno del PIC18F4550. De clic en **OK** para confirmar los datos.

Fuentes de señal e instrumentos de medición en PROTEUS

Una parte importante de la simulación de circuitos electrónicos, sean estos analógicos y/o digitales, radica en la visualización de las señales resultado de las fuentes que se le apliquen al circuito. Proteus cuenta con una gran variedad tanto de fuentes de señal como de instrumentos de medición. Las fuentes de señal son variadas, tales como fuentes de corriente directa, senoidales, exponenciales, de pulsos, etc. Estas se encuentran en el modo generador de operación (***Generator Mode***), en la barra de herramientas en el icono  . La manera de emplearlas es similar al manejo de un componente, debe ser agregado y conectado en los puntos necesarios del circuito.

Similarmente, los instrumentos de medición se encuentran en el modo de operación de instrumentos virtuales *(**Virtual Instruments Mode***) en la barra de herramientas en el icono  , de igual forma que las fuentes de señal y los dispositivos deben de agregarse y conectarse en los puntos a medir en el diagrama esquemático.

7. Posicione el osciloscopio dentro del diagrama esquemático del circuito, seleccionando el modo de operación de instrumentos virtuales. Realice la conexión de los mismos a los pines menos significativos del puerto B del PIC18F4550, el diagrama deberá lucir como lo muestra la figura 2.8.

En el caso del ejemplo desarrollado en este experimento, el voltaje de alimentación del MCU viene por defecto ya implícito, por lo que no se agregará fuente de señal alguna al diagrama esquemático.

Una vez “descargado” el programa ejecutable al MCU, colocadas todas las fuentes de señal y los instrumentos de medición en el circuito se procede a simularlo, para esto se emplearán los controles de simulación de la barra de herramientas, representados por los iconos  .

La función de los controles de simulación se describe a continuación:

PLAY  : Su pulsación hace que se inicie la simulación, cuando se está simulando cambia a color verde, mostrándose además, el tiempo que se lleva simulando y la carga computacional de CPU.

PAUSE : Si nos encontramos en el modo PLAY su pulsación hace que la simulación se detenga, la tecla de pausa cambia de color, y en la barra de simulación, se nos indica el tiempo transcurrido desde que se inició la simulación hasta que ésta ha sido detenida. Una nueva pulsación de esta tecla hará que la simulación se reanude en modo continuo.

STEP : Si nos encontramos en el modo PLAY su pulsación hace que la simulación se detenga, la tecla de pausa cambia de color, y en la barra de simulación, se nos indica el tiempo transcurrido desde que se inició la simulación hasta que esta ha sido detenida. Una nueva pulsación de esta tecla hará que la simulación se reanude, hasta que deje de presionarse o durante el tiempo especificado en las opciones de animación, es decir la simulación se hace paso a paso.

STOP : Si nos encontramos en el modo PLAY su pulsación hace que la simulación se detenga, saliendo el programa del modo simulación.

CPU LOAD: nos indica el porcentaje de utilización del CPU, en aquellas simulaciones/animaciones en las que dicho porcentaje se acerque al 100%, la simulación no se estará realizando en tiempo real.

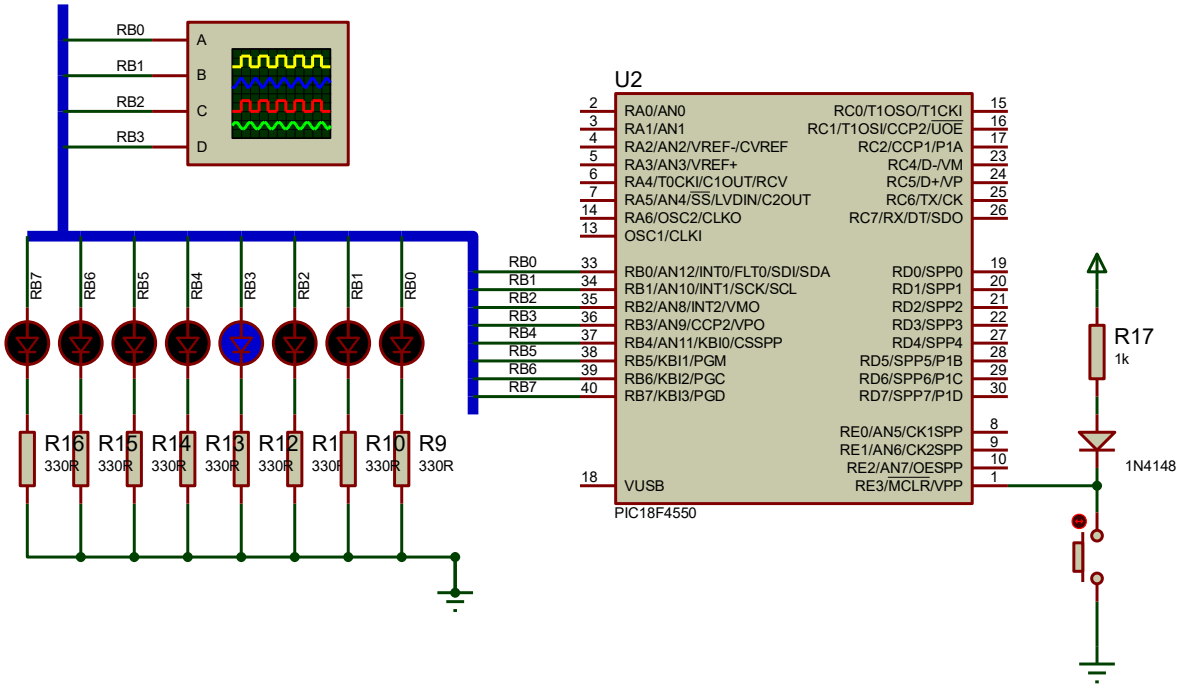


Figura 2.8: Conexión del osciloscopio en el diagrama esquemático.

8. Presione **PLAY** , en este momento la carátula del osciloscopio aparecerá en su pantalla, adicionalmente la barra de LEDS empezará a encender.

9. Ajuste la escala de tiempo de los canales del osciloscopio para que pueda visualizar las señales como lo muestra la figura 2.9. Cabe mencionar que el uso de los controles del osciloscopio es similar al de un osciloscopio real, incluyendo los controles de calibración.

Un punto importante que cabe mencionar en las simulaciones que involucran señales digitales, es la representación de los valores de los estados lógicos correspondientes. Proteus representa los valores lógicos de "1" a través de indicadores rojos en los pines que correspondan a salidas digitales; y con un indicador azul , los niveles lógicos de "0".

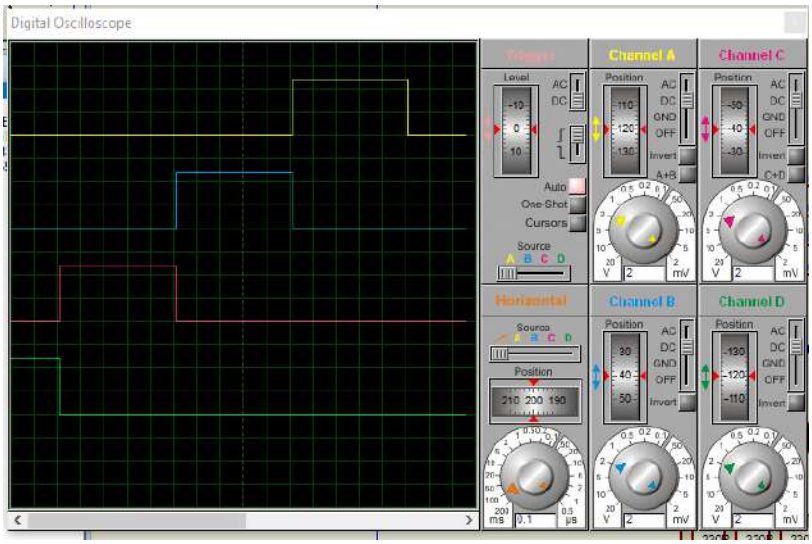


Figura 2.9: Simulación animada del diseño y visualización del osciloscopio.

De igual forma otro punto importante a tomar en cuenta es el hecho que si la simulación está en curso no podrán hacerse cambios en el diagrama esquemático. Adicionalmente si se cierran las carátulas de interfaz de las fuentes de señal o los instrumentos de medición en el transcurso de la simulación deberán ser eliminados del diagrama esquemático y volver a hacer las conexiones para que la interfaz del instrumento vuelva a mostrarse en la pantalla.

10. En relación con la implementación realizada en la práctica anterior, ¿Cómo es el comportamiento del circuito en PROTEUS?

_____

_____

11. Realice cambios en los controles que muestra el osciloscopio y presione el interruptor de "*reset*" del MCU. Observe y concluya.

12. Guarde los cambios en el diseño y cierre el programa **ISIS** y vuélvalo a abrir, abra el archivo correspondiente al ejercicio de esta práctica. De presentarse problemas al realizar este procedimiento, tal como el cierre repentino de **ISIS**, acceda a la carpeta donde guardó el archivo del proyecto y elimine todos los archivos de nombre similar, pero con extensión diferente. Este problema puede presentarse en versiones anteriores a la versión 7.8 de Proteus.

III. Simulación/depuración interactiva en Proteus.

El programa de diseño PROTEUS aparte de ser una herramienta poderosa y útil de diseño, permite un modo para simulación/depuración interactiva que proporciona varias funcionalidades útiles para la puesta a punto de diseños basados en MCU PIC; así como para la ayuda de la visualización del resultado de la ejecución de las instrucciones y modificando el circuito editado en PROTEUS.

13. Compile nuevamente su proyecto y visualice que en la carpeta donde lo tiene almacenado, deberá tener un nuevo archivo con extensión COF aparte del HEX generado previamente.

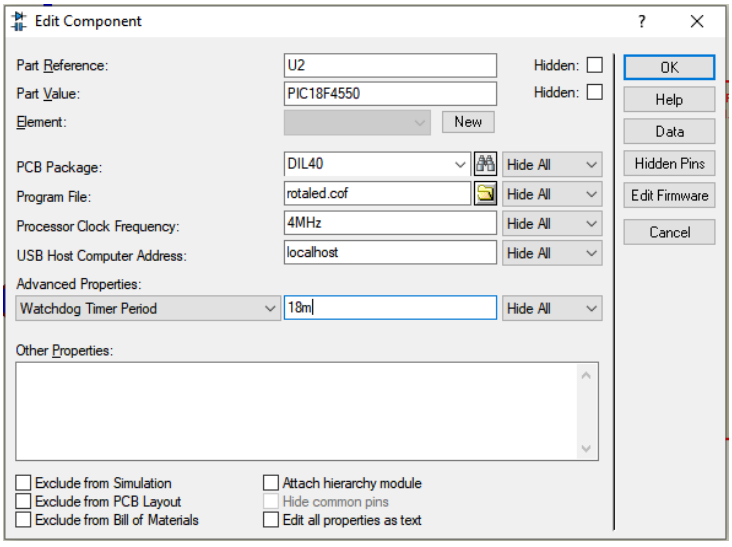


Figura 2.10: Ventana *Edit Component* para la "descarga" del archivo rotaled.cof al PIC18F4550.

14. Ejecute PROTEUS y repita el paso 6, pero ahora seleccione el archivo **rotaled.cof** en lugar del archivo **rotaled.hex**, tal como lo sugiere la figura 2.10.

15. Inicie la simulación del circuito con el botón “play” ubicado en la esquina inferior izquierda y verifique que efectivamente se esté desplazando un bit cada medio segundo. Luego, pulse el botón . A continuación, en el menú ***Debug > 4. PIC18 CPU***, despliegue las opciones de visualización de recursos internos del MCU correspondientes *a* ***Source Code, Registers***. Si todo sale bien, deberán desplegarse ventanas similares a las mostradas en la figura 2.11 y 2.12.

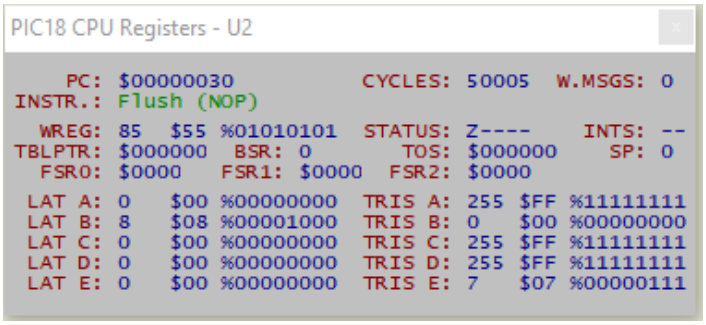


Figura 2.11: Ventana de visualización de los registros internos del PIC

16. La ventana de código fuente (figura 2.12), ofrece una serie de opciones para visualizar la ejecución del programa y al mismo tiempo ver su resultado gráfico en PROTEUS. Presione el icono , ¿qué fue lo que sucedió?

_____

¿De qué forma es posible volver a desplegar la ventana de la figura 2.12?

_____

_____

17. Con la ayuda del icono ejecute el código paso a paso y visualizando el efecto en los leds de su circuito. Preste especial atención al momento de llegar a la siguiente línea de código.

```
rrncf   LATB, f, .0
```

¿Qué sucede en la animación?

_____

¿Existe una diferencia en la simulación al pasar también por la siguiente línea, primero usando el icono y luego con el de .'

```
rcall    RETARDO_VAR_MS
```

_____

_____

PIC18 CPU Source Code - U2

C:\Users\Eduardo\MPLABXProjects\rotaled_V2_0.X\rotaled_v2_p1.asm

```
---- ; *************************************************************
---- ; Sistemas Digitales III p 2016
---- ; Encender un LED
---- ; Práctica 1
---- ;
---- ; Fecha: 27/I/2020
---- ; Notas: Encender un LED en RB4 al presionar un botón en RB0
---- ;
---- ; Autor: ERA
---- ; *************************************************************
----    LIST    P = 18F4550
----    INCLUDE <P18F4550.INC>
----
---- ;*************************************************************
---- ;               BITS DE CONFIGURACION
---- ;*************************************************************
---- ;********       Configuracion del Oscilador     **********
----         CONFIG  FOSC = INTOSCIO_EC  ;Osc interno, RA6 como pin, USB usa Osc EC
----
---- ;********      Bits de configuración mas usados     **********
----         CONFIG  PWRT    = ON        ;PWRT habilitado
----         CONFIG  BOR     = OFF       ;Brown out deshabilitado
----         CONFIG  WDT     = OFF       ;Watchdog deshabilitado
----         CONFIG  MCLRE   = ON        ;MCLR como entrada de reset
----         CONFIG  PBADEN  = OFF       ;PB como entradas digitales
----         CONFIG  LVP     = OFF       ;Prog bajo voltaje apagado
----         CONFIG  DEBUG   = OFF
----         CONFIG  XINST   = OFF
----
---- ;*********      Bits de proteccion      *****************
----         CONFIG  CP0     = OFF       ;los bloques del codigo de programa
----         CONFIG  CP1     = OFF       ;no estan protegidos
---- ;        CONFIG  CP2     = OFF
---- ;        CONFIG  CP3     = OFF
----         CONFIG  CPB     = OFF       ;Sector Boot no esta protegido
---- ;        CONFIG  CPD     = OFF       ;La EEPROM no esta protegida
----
---- ;#define Led PORTB, 4
----
---- ;******** Definición de variables     *****************
----  ; Variables en la ACCESS RAM
----         CBLOCK 0x00
----         CANT_MS:2                   ;Variable para generar 65535 s
----         ;CABT_SEG                   ;Variable para retardos de hasta 255s
----         STATUS_TEMP                 ;Variable clon para el STATUS
----         WREG_TEMP                   ;Varioable clon para el WREG
----         ENDC
----
---- ;*************************************************************************
---- ; Vector de Reset.
----         ORG     0x0000
0000         goto    INICIO              ;Ve al inicio del programa principal
----
---- ;*************************************************************************
---- ; Vector de interrupcion de baja prioridad y rutina
----         ORG     0x0018
0018         goto    INICIO              ;Ve al inicio del programa principal
----
---- ;*************************************************************************
---- ; Subrutina que configura la base de tiempo del MCU
---- CONF_BASE_TIEMPO
```

Figura 2.12: Ventana de visualización de código fuente en ensamblador.

18. En la pantalla de simulación de Proteus presione el interruptor conectado al pin RE3/MCLR'/VPP. ¿Qué acción se muestra en la ventana de Proteus? ¿Qué valores toman las salidas del MCU? Concluya al respecto.

_____

_____

¿Qué sucede en la ventana de que muestra los registros del PIC?, ¿Hubo algún cambio?

_____

_____

19. Repita esta tarea y experimente cambiando la instrucción a la que hace referencia el inciso 17, con las siguientes mnemónicos: RLCF, RLNCF, RRCF. Describa las diferencias que se perciben en puerto B en cada una de ello.

_____

_____

_____

20. Agregue el instrumento virtual ***timer counter*** al circuito tal como se indica en la figura 2.13. Con su ayuda verifique el tiempo exacto del retardo en el desplazamiento del bit en el puerto B. Anote dicho dato y compárelo calculándolo de forma manual analizando el programa.

_____

_____

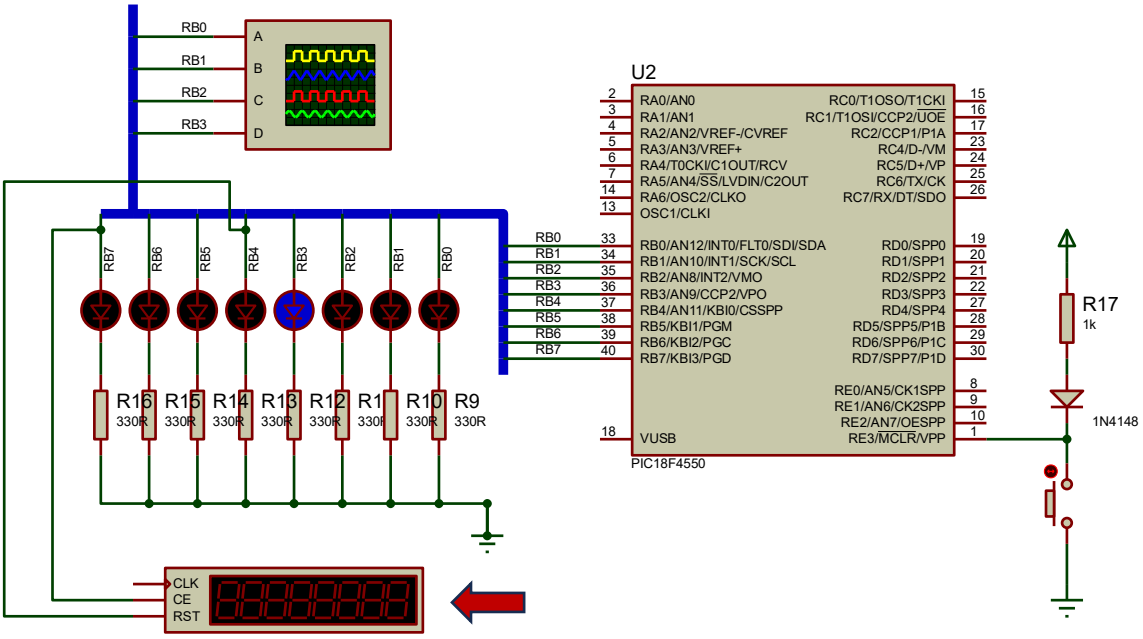


Figura 2.13: Circuito con instrumento ***timer counter*** incluido.

21. Escriba que otros recursos internos del MCU PIC en cuestión pueden ser visualizados en este modo de simulación/depuración interactiva de Proteus.

_____

_____

Note que este programa puede ser de suma utilidad al desarrollar aplicaciones que requieran múltiples pruebas y cambios en la estructura del programa o configuración de los módulos periféricos del microcontrolador. Sin embargo, tenga en cuenta que es un simulador, por lo que sus resultados están en función de los modelos proporcionados por el desarrollador del programa.

ACTIVIDADES COMPLEMENTARIAS

1. Investigue las propiedades de las fuentes de señal y los instrumentos virtuales con los que cuenta Proteus.

2. Experimente las funcionalidades extras que agrega Proteus para la depuración de circuitos basados en MCU.

3. ¿Cómo es el procedimiento para que en Proteus se pueden visualizar el valor que toman las variables durante la simulación:

4. Investigue la diferencia entre un emulador y un simulador.

TEMA 3

LOS BITS DE CONFIGURACIÓN DEL PIC18F4550

INTRODUCCIÓN

El PIC18F4550 de Microchip es un potente microcontrolador CMOS de 8 bits con arquitectura RISC capaz de operar con diversas frecuencias, fácil de programar (sólo 75 instrucciones) y disponible en diversos tipos de encapsulados.

El PIC18F4559 ha sido construido con características tales que se puede configurar para funcionar en modos de operación que no necesitan componentes externos tales como el circuito de reloj o de ***reset***. Esto implica que además de elaborar el programa que deseamos ejecutar, también es necesario configurar su modo de operación a través de varias palabras de configuración (***configuration word***). Dichas palabras se encuentran mapeadas desde la dirección 300000h a la 30000Dh de la memoria de programa y sólo se puede acceder durante la programación de dispositivo. Así, ya sea a través de directivas del MPASM incrustadas dentro del código fuente en ensamblador o en el menú ***production >> configuration*** bits de MPLAB X. Otra forma de modificarlos es durante el proceso de descarga y programación del archivo HEX al PIC, por medio de la interfaz del dispositivo programador empleado.

Tabla 3.1: Bits de configuración del PIC18F4550.

Dirección	Registro	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default
300000h	CONFIG1L	—	—	USBDIV	CPUDIV1	CPUDIV2	PLLDIV2	PLLDIV1	PLLDIV0	--00 0000
300001h	CONFIG1H	IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0	00—0101
300002h	CONFIG2L	—	—	VREGEN	BORV1	BORV0	BOREN1	BOREN0	PWRTEN’	--01 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	---1 1111
300005h	CONFIG3H	MCLRE	—	—	—	—	LPT1OSC	PBADEN	CCP2MX	1--- -011
300006h	CONFIG4L	DEBUG’	XINTS	ICPRT	—	—	LVP	—	STVREN	100- 1-1-
300008h	CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0	---- 1111
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—		WRT3	WRT2	WRT1	WRT0	---- 1111
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0	---- 1111
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	xxxx xxxx
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0001 0010

En la Tabla 3.1, se identifican en recuadro rojo alguno de los bits de configuración que serán objeto de experimentación en esta práctica. Su significado y uso se describen a continuación.

FOSC3:FOSC2:FOSC1:FSC0: Bits de selección del tipo de oscilador.

Tabla 3.2: Bits de selección del tipo de oscilador del PIC18F4550.

Bits	Tipo de oscilador	Abrev.
111X	Oscilador HS, PLL habilitado	HSPLL
110X	Oscilador HS	HS
1011	Oscilador interno, oscilador HS usado por el USB	INTHS
1010	Oscilador interno, oscilador XT usado por el USB	INTXT
1001	Oscilador interno, RA6 = CLK0, EC usado por el USB	INTCKO
1000	Oscilador interno, RA6 = pin de puerto, EC usado por el USB	INTIO
0111	Oscilador EC, RA6 = CLK0	ECPLL
0110	Oscilador EC, PLL habilitado,	ECPIO
0101	Oscilador EC, RA6 = CLK0	EC
0100	Oscilador EC, RA6 = pin de puerto	ECIO
001X	Oscilador XT, PLL habilitado	XTPLL
000X	Oscilador XT	XT

PBADEN: Bit de habilitación del PORTB como puerto digital.
(Nota: Este bit afecta el estado al *reset* del registro ADCON1)
1 = Los pines PORTB<4:0> son configurados como canales de entrada analógicos.
0 = Los pines PORTB<4:0> son configurados como pines de I/O digitales.

LVP: Habilitación de la programación por voltaje bajo.
1 = LVP habilitado, la terminal RB3/PGM tiene tal función. 0 = LVP deshabilitado, RB3 es una terminal de I/O.

MCLRE: Habilitación de la terminal de reset.
1 = Terminal de *reset* (MCLR) en RE3.
0 = MCLR conectado internamente a VDD, RE3 es un pin de I/O.

El experimento 3 tiene como objetivos principales:

- Identificar algunos de los bits de configuración relevantes que rigen el funcionamiento del PIC18F4550.
- Verificar funcionalmente el efecto de los bits de configuración.
- Configurar los puertos paralelos correctamente del PIC18F4550.

Se necesita contar con el siguiente equipo:

- Computadora con los programas Proteus ® versión 7.9 o superior y MPLAB® X v5.0 o superior instalados.
- Dispositivo programador compatible con PIC18F4550.
- Osciloscopio.
- Generador de funciones.
- Fuente de alimentación de CD.

Así mismo, se enlista los materiales a utilizar durante el desarrollo del experimento 3:

Cantidad	Descripción
1	Microcontrolador PIC18F4550
1	Microinterruptor (1P-1T) óctuple.
8	Resistencia de 330 Ω
1	Resistencia de 3.3 kΩ
4	Resistencia de 4.7 kΩ
1	*Display* de cátodo común.
1	Oscilador de 8 MHz.
1	Cristal de 8 MHz.
1	Cristal de 4 MHz.
2	Capacitores de 27 pF.
2	Capacitor 33 pF.
1	Push-button.
1	Tableta experimental

Se recomienda leer previamente todo el documento del experimento 3 y llevar implementado en un *protoboard* el circuito de la figura 3.10.

EXPERIMENTO 3

DESARROLLO

I. Bits de configuración del tipo de oscilador.

El PIC18F4550 posee diversas formas de configurar al oscilador tanto para su operación como para la del puerto USB que trae integrado. Un oscilador de tipo LP es usado en aplicaciones de bajo consumo. El modo XT es el más empleado y usa un cristal. El HS emplea cristales de alta velocidad o resonadores cerámicos. Por último, el modo oscilador interno INTIO puede elegirse para operar a alta o baja velocidad. Estas configuraciones hacen más flexibles las aplicaciones con este microcontrolador.

1.- Editar el programa en ensamblador que se muestra en la figura 3.2 y asígnele el nombre de **decodifica.asm**.

```
; ********************************************************************
; Sistemas Digitales III p 2016
; Decoder de 3 a 7 segmentos con display de catodo común
; Práctica 3
;
; Fecha: 27/I/2020
; Autor: ERA
; ********************************************************************
      LIST        P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX H     EX

;*******************************************************************
;               BITS DE CONFIGURACION
;*******************************************************************
;********       Configuracion del Oscilador     ***********
      CONFIG  FOSC = INTOSCIO_EC    ;Osc interno, RA6 como pin libre

;********      Bits de configuración mas usados     ******************
      CONFIG  PWRT    = ON          ;PWRT habilitado
      CONFIG  BOR     = OFF         ;Brown out deshabilitado
      CONFIG  WDT     = OFF         ;Watchdog deshabilitado
      CONFIG  MCLRE   = ON          ;MCLR como entrada de reset
      CONFIG  PBADEN  = OFF         ;PB como entradas digitales
      CONFIG  LVP     = OFF         ;Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF         ;los bloques del codigo de programa
      CONFIG  CP1     = OFF         ;no estan protegidos
      CONFIG  CPB     = OFF         ;Sector Boot no esta protegido

;*******************************************************************
; Vector de Reset.
      ORG   0x0000
      goto  INICIO            ;Ve al inicio del programa principal

;*******************************************************************
; Vector de interrupcion de alta prioridad
      ORG   0x0008
      goto  INICIO            ;Ve al inicio del programa principal
```

Figura 3.1: Código fuente en ensamblador para el circuito experimental con el PIC18F4550.

```
;*********************************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
      movlw   B'01100010'
      movwf   OSCCON          ;Oscilador interno a 4 MHz
      return

;*********************************************************************
; Subrutina que configura todo el puerto B como puerto de salida y
; el puerto A como entrada digital. Se deshabilitan comparadores
CONF_PUERTOS
      clrf    LATB, 0         ;Limpia Lacths del puerto B
      clrf    LATA, 0         ;Limpia Lacths del puerto A
      movlw   0Fh
      movwf   ADCON1, 0       ;Todos los pines como I/O digitales
      clrf    TRISB, 0        ;Todos loe pines de puerto B como salidas
      movlw   0FFh
      movwf   TRISA, 0        ;Todos el puerto A como entradas
      clrf    LATB, 0         ;Puerto B = 0000 0000b
      return

;*********************************************************************
; Subrutina que lee el puerto A y enmascara los 3 LSB
LEE_PUERTO_A
      movf    PORTA, W, .0    ;Lee valor de pines del puerto A
      andlw   b'00000111'           ;Enmascara los 3 LSB
      rlncf   WREG, W, .0
      return

;*********************************************************************
; Subrutina que decodifica el valor de WREG a su correspondiente en 7
segmentos, se asume display de cátodo común y la conexión de los segmentos
con los pines del puerto B como:; RB0-a, RB1-b, ... ,RB6-g, RB7-punto.
DECODIFICA
      addwf   PCL, f, .0
      retlw   b'00111111'
      retlw   b'00000110'
      retlw   b'01011011'
      retlw   b'01001111'
      retlw   b'01100110'
      retlw   b'01101101'
      retlw   b'01111101'
      retlw   b'00000111'
```

Figura 3.1: Código fuente en ensamblador para el circuito experimental con el PIC18F4550 (cont.)

```
;*****************************************************************
;       PROGRAMA PRINCIPAL
INICIO
      rcall   CONF_BASE_TIEMPO
      rcall   CONF_PUERTOS
AGAIN
      Rcall   LEE_PUERTO_A
      rcall   DECODIFICA
      movwf   LATB, 0
      bra     AGAIN

      END
```

Figura 3.1: Código fuente en ensamblador para el circuito experimental con el PIC18F4550 (cont.)

2. Al circuito solicitado en las actividades previas, agregar la parte del reloj con cristal (XTAL) de 8 MHz, tal como se muestra en la figura 3.2.

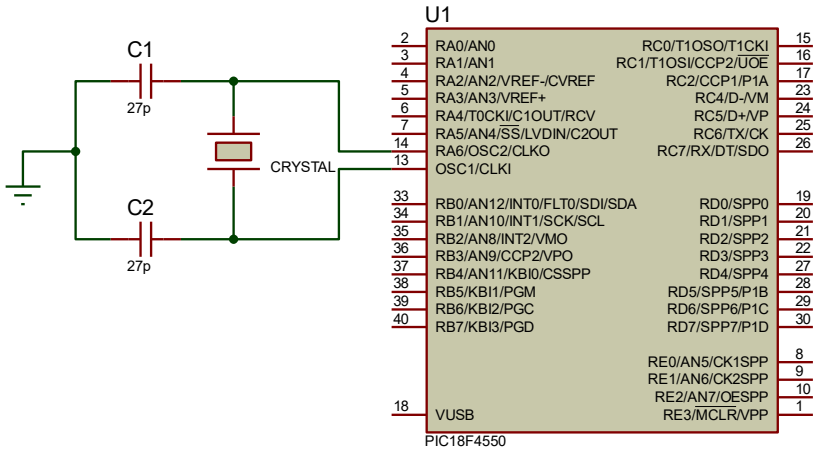


Figura 3.2: Circuito de reloj con cristal (XTAL).

3. En MPLAB X, es posible editar los bits de configuración y generar automáticamente el código ASM para tal fin. Para esto, navegar al menú ***Production > Set configuration Bits***, deberá aparecer una ventana similar a la mostrada en la figura 3.3. Edite únicamente el bit de configuración del PIC18F4550 referente a la frecuencia de oscilación para que tenga la opción XT_XT y de *click* en el botón ***Generate Source Code to Ouput***.

¿Identifique y escriba la línea de código que genera MPLAB X y que configura el tipo de oscilador?

_____

¿Cuál es el nombre del registro donde se refleja dicha modificación? _____

4. Energice su circuito, ¿funciona?

_____

¿A qué modo de oscilador corresponde la opción configurada?

_____

Es muy probable que su circuito no funcione, explique ¿por qué?

_____

_____

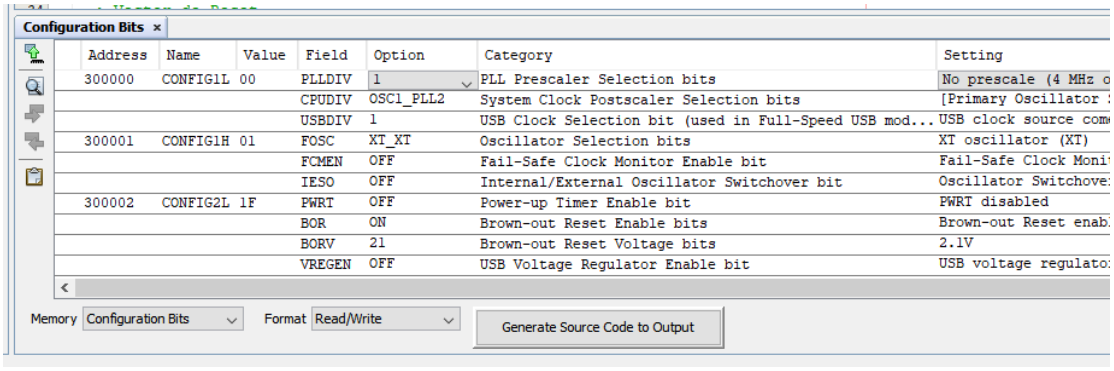


Figura 3.3: Ventana de bits de configuración de MPLAB X.

5. Modifique la configuración del oscilador a la opción ***HS oscillator*** (ver figura 3.4), con la ayuda MPLAB, genere el código correspondiente y modifique su archivo ASM.

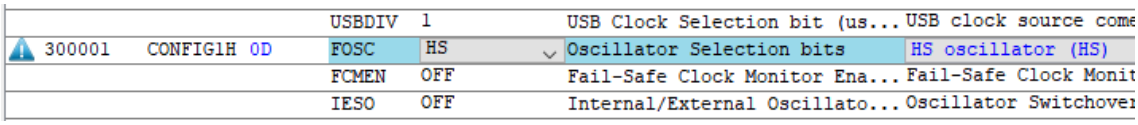


Figura 3.4: Bits de configuración para el inciso 5.

Escriba como debe quedar la línea de código con la modificación que se plantea.

_____

¿Hubo algún cambio en el valor del registro CONFIG1H? _____

¿A qué modo de oscilador corresponde esta palabra de configuración? _____

¿Funciona el circuito?_____
Explique:_____

_____

6. Sustituya en el circuito el XTAL de 8 MHz por uno de 4 MHz y los capacitores de 27 pF por los de 22 pF y repita el paso 5. ¿Funcionó el circuito?, Explique:

_____

_____

7. Modifique su circuito como lo muestra la figura 3.5 y reprograme su PIC seleccionando la palabra de configuración mostrada en la figura 3.6.

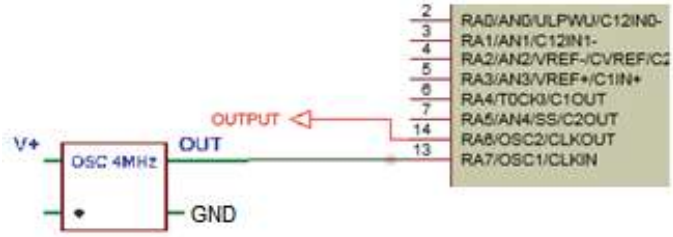


Figura 3.5: Circuito de reloj con oscilador externo.

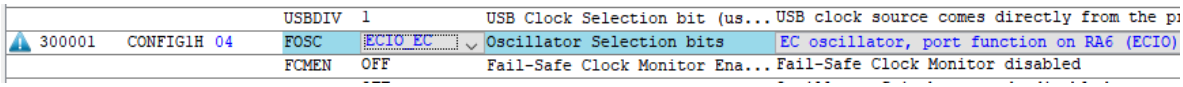


Figura 3.6: Palabra de configuración para el circuito de la figura 3.5, paso 7.

8. Verifique el funcionamiento y explique, ¿Qué modo de oscilador indica la palabra de configuración?

_____

_____

Monitoree con un canal del osciloscopio la salida OUTPUT (PIN 14-RA6), ¿Se observa alguna señal?

_____

9. Repita el paso anterior, pero ahora con la configuración mostrada en la figura 3.7.

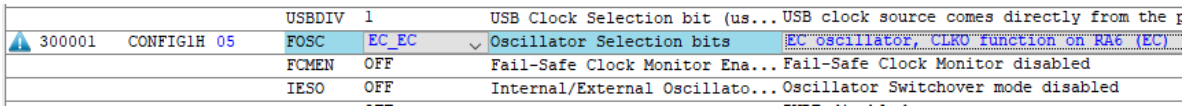


Figura 3.7: Palabra de configuración para el circuito de la figura 3.5, paso 9.

¿Existe alguna diferencia en funcionamiento con respecto a la configuración del paso 7?, Describa y explique el ¿por qué de este comportamiento?

_____

¿Para qué sería útil emplear este tipo de configurar el MCU?

_____

Con la ayuda del osciloscopio, determine ¿cuál es la frecuencia de la señal?

_____

¿A qué se debe esto?

_____

_____

10. El PIC18F4550 tiene la capacidad de generar internamente las señales para su funcionamiento. ¿Cuál de las opciones mostradas en la tabla de osciladores debe elegirse en la ventana de MPLAB X de la figura 3.4?

_____

Modifique su código, reprograme su PIC y compruebe experimentalmente lo anterior.

11. Existirá en este modelo del PIC, una opción semejante a la del paso 9?_____
En caso de haberla ¿cuál de las opciones correspondería?

_____

Reprograme su PIC con la opción mencionada y compruebe con el osciloscopio si existe señal alguna por el pin RA6. ¿Cuál es su frecuencia?_____

Analice su código y ubique la subrutina donde configura la frecuencia interna de oscilación, cambie el valor en el registro correspondiente para que opere a 8 MHz, reprograme su PIC y verifique si hubo cambio en la señal del osciloscopio. Concluya al respecto.

_____

II. Bit de configuración del *reset* (MCLR)

Cuando la terminal MCLR/VPP (*Master Clear*) tiene un nivel de tierra, todos los registros del PIC se ponen en estado conocido o en estado de *reset*. El arreglo típico implica conectar una resistencia de 10kΩ a 5V, un diodo y un pulsador a tierra (GND), de tal forma que cuando se presiona el pulsador el MCU se reinicializa.

11. Reprograme el PIC con las opciones mostradas en la ventana de la figura 3.8. Cabe señalar que a partir de este punto y de lo que resta de la práctica, se usará la configuración de reloj interno del MCU a 4 MHz.

Oprima el botón del *reset* del circuito ¿qué es lo que sucede?

_____

¿Por qué?

_____

```
;*****************************************************************
;               BITS DE CONFIGURACION
;*****************************************************************
;********       Configuracion del Oscilador     ***********
      CONFIG  FOSC = INTOSCIO_EC    ;Osc interno, RA6 como pin libre

;********      Bits de configuración mas usados     ******************
      CONFIG  PWRT    = ON          ;PWRT habilitado
      CONFIG  BOR     = OFF         ;Brown out deshabilitado
      CONFIG  WDT     = OFF         ;Watchdog deshabilitado
      CONFIG  MCLRE   = ON          ;MCLR como entrada de reset
      CONFIG  PBADEN  = OFF         ;PB como entradas digitales
      CONFIG  LVP     = OFF         ;Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF         ;los bloques del codigo de programa
      CONFIG  CP1     = OFF         ;no estan protegidos
      CONFIG  CPB     = OFF         ;Sector Boot no esta protegido
```

Figura 3.8: Bits de configuración para el paso 11.

12. Modifique la opción mostrada en la figura 3.8 y deshabilite la opción MCLR, reprograme su PIC. Repita el paso 10, ¿diga qué fue lo que sucedió?

_____

¿Por qué?

_____

_____

¿Es posible “resetear” el circuito? Explique cómo.

_____

_____

III. Bit de configuración LVP

El modo de programación de bajo voltaje (LVP) invalida el empleo de alto voltaje para la programación del PIC. Aquí, el dispositivo puede ser programado sin usar 12V de voltaje de programación. Sin embargo, cuando se utiliza la programación del alto voltaje mientras que el MCU tiene activada la de la baja tensión, éste último modo no se elimina. De tal forma si RB5 pasa a alto por cualquier razón durante la programación con alto voltaje, la programación se interrumpirá.

```
;*******************************************************************
;                BITS DE CONFIGURACION
;*******************************************************************
;********       Configuracion del Oscilador     ***********
      CONFIG  FOSC = INTOSCIO_EC    ;Osc interno, RA6 como pin libre

;********      Bits de configuración mas usados     ******************
      CONFIG  PWRT    = ON          ;PWRT habilitado
      CONFIG  BOR     = OFF         ;Brown out deshabilitado
      CONFIG  WDT     = OFF         ;Watchdog deshabilitado
      CONFIG  MCLRE   = ON          ;MCLR como entrada de reset
      CONFIG  PBADEN  = OFF         ;PB como entradas digitales
      CONFIG  LVP     = ON          ;Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF         ;los bloques del codigo de programa
      CONFIG  CP1     = OFF         ;no estan protegidos
      CONFIG  CPB     = OFF         ;Sector Boot no esta protegido
```

Figura 3.9: Palabra de configuración para el inciso 13.

13. Programe su PIC modificando la línea de código resaltada en rojo en la figura 3.9.

14. Energice su circuito y verifique ¿funciona? _____

Por qué?

_____

_____

15. ¿De qué forma se puede corregir esta anomalía?

_____

_____

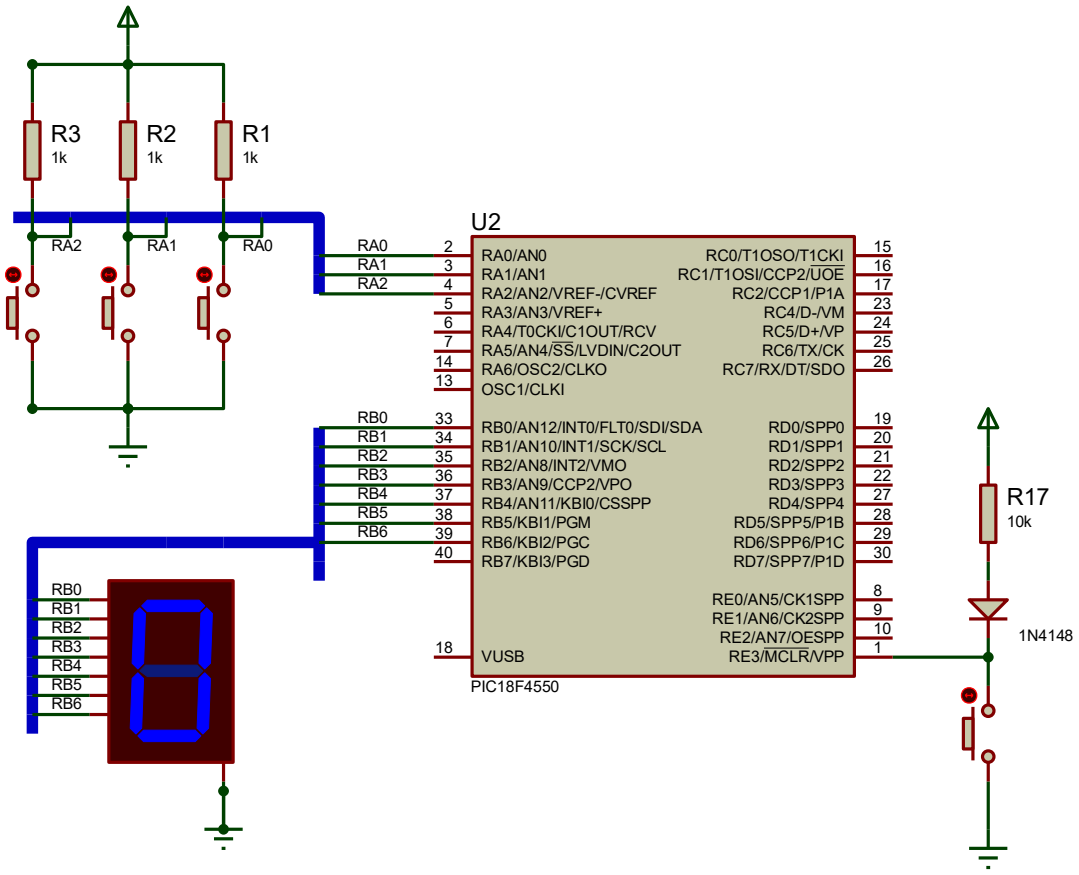


Figura 3.10: Circuito decodificador de 7 segmentos con el PIC18F4550.

ACTIVIDADES COMPLEMENTARIAS

1. Investigue las principales diferencias entre el PIC16f887 y su antecesor el PIC18F4550.

2. Investigue que bits de configuración adicionales existen en el PIC18F4550 y que no tiene el PIC16F887. Asimismo, describa brevemente su funcionalidad.

3. Investigue para qué sirve el bit de configuración ***Watchdog Timer*** y explique un ejemplo de su posible uso en una aplicación.

TEMA 4

DIRECCIONAMIENTO INDIRECTO EN EL PIC18F4550

INTRODUCCIÓN

En los microcontroladores PIC, la mayoría de las instrucciones emplean **direccionamiento directo**, lo que significa que la dirección de la memoria donde se encuentran los datos se especifica de manera explícita en la instrucción. Sin embargo, **el direccionamiento indirecto** permite acceder a una ubicación de la memoria de datos sin especificar una dirección fija, lo que brinda mayor flexibilidad en el manejo de datos.

Para lograr el direccionamiento indirecto, se utilizan registros especiales (FSRxH y FSRxL) llamados **FSR** (*File Select Registers*), los cuales funcionan como apuntadores a una dirección específica en la memoria de datos. En lugar de referirse directamente a una dirección de memoria en la instrucción, el programa modifica el valor de estos registros para acceder dinámicamente a diferentes posiciones de memoria.

Otro conjunto de registros esenciales en este esquema son los **INDF** (*Indirect File Registers*). Los INDF no existen físicamente como registros en el hardware, sino que representan el contenido de la dirección apuntada por el registro FSR correspondiente. Esto significa que cuando una instrucción usa un registro INDF, en realidad está accediendo al valor almacenado en la dirección apuntada por el registro FSR.

Algunas de las ventajas del direccionamiento indirecto son:

- **Auto (pre/pos) incremento y auto (pre/pos) decremento:** Los registros FSR pueden modificar automáticamente su valor después de cada acceso, facilitando el recorrido secuencial de datos sin necesidad de actualizar manualmente la dirección de memoria.
- **Uso de Offset:** Se pueden combinar valores de desplazamiento (*offset*) con el valor del apuntador para acceder a ubicaciones relativas dentro de la memoria
- **Elimina el uso de bancos de memoria:** Debido a que el direccionamiento indirecto usa direcciones de 12 bits, el acceso a memoria de datos en forma de bancos no es necesario. De esta forma, los contenidos actuales de los registros BSR's (*Bank Select Register*) y de la *access RAM* no tienen ningún efecto para determinar la ubicación objetivo.

En resumen, los registros FSR's sirven para "señalar o apuntar" a una dirección cuyo contenido después puede ser leído ó escrito de forma indirecta empleando cualquier instrucción que use como operando a alguno de los registros INDF's. La figura 4.1 muestra un ejemplo de una instrucción que usa direccionamiento indirecto.

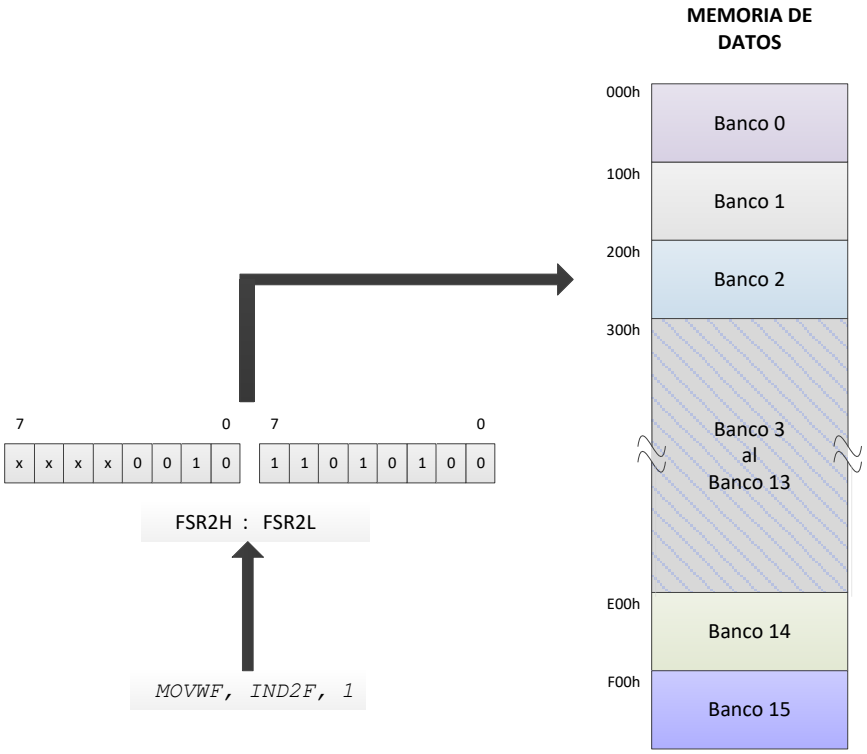


Figura 4.1: Ejemplo de direccionamiento indirecto.

De tal forma, que esta forma de direccionamiento es particularmente útil cuando se manejan tablas o arreglos en la memoria de datos del PIC.

Con el experimento 4 se logrará identificar los registros involucrados en el modo de direccionamiento indirecto y analizarás a través del entorno de simulación del MPLAB, la forma de operar del modo de direccionamiento indirecto.

Para realizar este experimento es necesario contar con una computadora con los programas PROTEUS versión 7.9 o superior y MPLAB® X v5.0 o superior instalados.

Se recomienda leer previamente todo el documento del experimento 4 e investigar ¿cuáles son los registros que se usan para el direccionamiento indirecto?

EXPERIMENTO 4

DESARROLLO

1. Edite el siguiente programa con MPLAB IDE (no MPLAB X) y genere el proyecto de la forma acostumbrada. Ensamble y depure hasta que no existan errores o *warning*s.

```
; *************************************************************
; Sistemas Digitales III p 2016
; Programa que realiza la suma de 10 números almacenados
; a partir de la dirección 23h de la memoria de datos
; Fecha: 27/I/2020
; Autor: ERA
; *************************************************************
      LIST        P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX       HEX

;*************************************************************
;               BITS DE CONFIGURACION
;*************************************************************
;********       Configuracion del Oscilador     **********
;       Osc interno, RA6 como pin, USB usa Osc EC
      CONFIG  FOSC    = INTOSCIO_EC

;********      Bits de configuración mas usados     **********
      CONFIG  PWRT    = ON    ; PWRT habilitado
      CONFIG  BOR     = OFF   ; Brown out deshabilitado
      CONFIG  WDT     = OFF   ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON    ; MCLR como entrada de reset
      CONFIG  PBADEN  = OFF   ; PB como entradas digitales
      CONFIG  LVP     = OFF   ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF   ; los bloques del codigo de programa
      CONFIG  CP1     = OFF   ; no estan protegidos
      CONFIG  CPB     = OFF   ; Sector Boot no esta protegido
```

Figura 4.2: Código fuente en ensamblador para el experimento del punto 1.

```
RMM    MACRO       BYTES
       ORG         $+BYTES
       ENDM

       CBLOCK      00h
       SUMAH
       SUMAL
       LONG
       ENDC

       ORG   20h
TABLA RMM   .20

;*********************************************************************
; Vector de Reset.
       ORG   0x0000
       goto  INICIO          ; Ve al inicio del programa principal

;*********************************************************************
; Vector de interrupcion
       ORG   0x0008
       goto  INICIO          ; Ve al inicio del programa principal

;*********************************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
       movlw   B'01100010'
       movwf   OSCCON          ; Oscilador interno a 4 MHz
       return

;*********************************************************************
;       PROGRAMA PRINCIPAL
INICIO
       rcall   CONF_BASE_TIEMPO
       clrf    SUMAH, 0        ; Suma total parte alta = 00h
       clrf    SUMAL, 0        ; Suma total parte baja = 00h
       movlw   .10             ; Cantidad de números
       movwf   LONG, 0         ; a sumar = 10
       lfsr    FSR0, TABLA     ; Coloca apuntador al inicio de la tabla

OTRO
       movf    SUMAL, w, 0     ; w <-- SUMAL
       addwf   INDF0,w, 0      ; w <-- SUMAL+(INDF)
       movwf   SUMAL, 0
       btfsc   STATUS, C, 0    ; Hubo acarreo?
       incf    SUMAH, f, 0     ; SI, SUMAH ++
       incf    FSR0L, f, 0     ; Apunta al sig. elemento de la tabla
       decfsz  LONG, f, 0      ; Es el último elemento?
       bra     OTRO            ; No, continua sumando
       bra     $               ; Si, termina

       END
```

Figura 4.2: Código fuente en ensamblador para el experimento del punto 1 (cont.)

2. Elabore el diagrama de flujo correspondiente al programa.

3. Proceda a inicializar la tabla con los valores a sumar, con el menú ***view>> 4 file registers***, lo cual desplegará una ventana donde se puede modificar el contenido de los GPR's. En nuestro caso vamos a introducir los valores que se muestran en la figura 4.3.

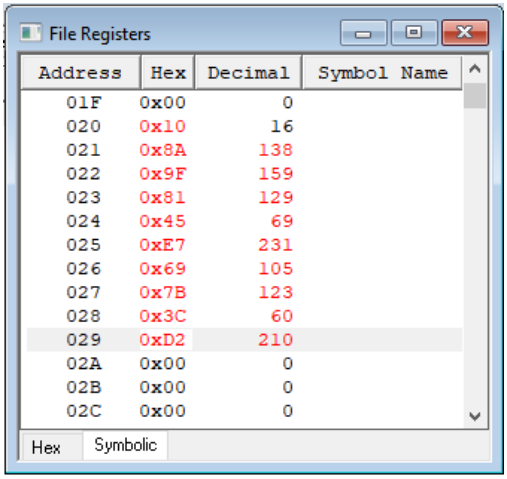


Figura 4.3: Ventana que muestra el contenido de las direcciones 20h a 29h.

4. Inserte un punto de ruptura (***BREAK-POINT***) en la instrucción ***decfsz LONG, f, 0*** (Para insertar un punto de ruptura, dar doble "click" con el ratón posicionado en la instrucción, para borrarlo dar doble "click" nuevamente).

5. Active el simulador MPLAB SIM en ***Debugger>>Select tool>>MPLAB SIM*** y simule un ***RESET*** al microcontrolador, con el botón , a continuación, ejecute la simulación del programa con el botón ***RUN*** . Escriba en el primer renglón de la tabla 4.1, el valor que tienen las siguientes variables cuando la simulación se detiene al llegar al punto de ruptura.

6. Reanude la simulación y vuelva anotar el contenido de los registros. Repita este paso hasta que el programa finalice. Registre sus resultados en la tabla 4.1.

Tabla 4.1: Valor de las variables del programa durante su ejecución.

	SUMAH	**SUMAL**	**FSR0**	**LONG**
Pasada 1				
Pasada 2				
Pasada 3				
Pasada 4				
Pasada 5				
Pasada 6				
Pasada 7				
Pasada 8				
Pasada 9				
Pasada 10				

7. Modifique el contenido de la tabla para que contenga “FFh” en todas sus localidades. Elabore una tabla similar a la anterior.

Tabla 4.2: Valor de las variables del programa (con la tabla llena de FFh).

	SUMAH	**SUMAL**	**FSR0**	**LONG**
Pasada 1				
Pasada 2				
Pasada 3				
Pasada 4				
Pasada 5				
Pasada 6				
Pasada 7				
Pasada 8				
Pasada 9				
Pasada 10				

8. Analice el programa y explique ¿De qué forma se puede incrementar la cantidad de números que se pueden sumar?

_____

9. ¿Para qué sirve la macro RMM?

_____

_____

10. Dentro de la macro ubique la línea de código.

```
ORG   $+BYTES
```

¿Qué lo que hace exactamente dentro de la macro?

_____

11. Explique, ¿por qué no fue posible utilizar la directiva **RES** para reservar los bytes necesarios para la etiqueta **TABLA**?

_____

_____

12. Dentro del programa principal ubique la línea de código.

```
BRA       $
```

¿Qué es lo que hace exactamente?

_____

¿Qué sucede en el programa si se omite?

---

13. Del código del punto 1, modifique ahora únicamente el programa principal en la forma que se muestra en el listado de abajo.

```
;*****************************************************************************
;       PROGRAMA PRINCIPAL
INICIO
      rcall   CONF_BASE_TIEMPO
      clrf    SUMAH, 0          ; Suma total parte alta = 00h
      clrf    SUMAL, 0          ; Suma total parte baja = 00h
      movlw   .10               ; Cantidad de números
      movwf   LONG, 0           ; a sumar = 10
      lfsr    FSR0, TABLA       ; Coloca apuntador
                                ; al inicio de la tabla

OTRO
      movf    SUMAL, w, 0       ; w <-- SUMAL
      addwf   PREINC0,w, 0      ; w <-- SUMAL+(INDF)
      movwf   SUMAL, 0
      btfsc   STATUS, C, 0      ; Hubo acarreo?
      incf    SUMAH, f, 0       ; SI, SUMAH ++
      decfsz  LONG, f, 0        ; Es el último elemento?
      bra     OTRO              ; No, continua sumando
      bra     $                 ; Si, termina

      END
```

Figura 4.2: Modificación del código fuente en ensamblador para el experimento del punto 13.

14. Repita los pasos necesarios para inicializar la memoria nuevamente con los valores indicados en la figura 4.3, ensamble y simule paso a paso el programa. Actualice la tabla 4.3 con los valores obtenidos.

Tabla 4.3: Valor de las variables para el programa del punto 14.

	SUMAH	SUMAL	FSR0	LONG
Pasada 1				
Pasada 2				
Pasada 3				
Pasada 4				
Pasada 5				
Pasada 6				
Pasada 7				
Pasada 8				
Pasada 9				
Pasada 10				

15. Compare las tablas 4.1 y 4.3. ¿Existe o no alguna discrepancia en cada uno de los valores?

_____

Explique, ¿a qué se debe tal comportamiento?

_____

_____

16. En el código del punto 12, ubique la instrucción ***addwf PREINC0, w, 0*** y sustitúyala con ***addwf POSTINC0, w, 0***, repita los puntos 14, 15 y anote sus resultados en la tabla 4.4. Describa, ¿cuál es ahora el comportamiento en relación con los anteriores? Concluya al respecto.

_____

_____

Tabla 4.4: Valor de las variables para el programa del punto 16.

	SUMAH	**SUMAL**	**FSR0**	**LONG**
Pasada 1				
Pasada 2				
Pasada 3				
Pasada 4				
Pasada 5				
Pasada 6				
Pasada 7				
Pasada 8				
Pasada 9				
Pasada 10				

17. Modifique la tabla nuevamente con los valores mostrados en la figura 4.3. De igual forma reescriba su código ASM de manera tal, que ahora el recorrido de los valores de la memoria se haga en forma inversa, es decir iniciando con el último valor y terminando con el primero. Ejecute el programa y llene la tabla 4.5

Tabla 4.5: Valor de las variables para el programa del punto 16.

	SUMAH	SUMAL	FSR0	LONG
Pasada 1				
Pasada 2				
Pasada 3				
Pasada 4				
Pasada 5				
Pasada 6				
Pasada 7				
Pasada 8				
Pasada 9				
Pasada 10				

ACTIVIDADES COMPLEMENTARIAS

1. Investigue 3 ventajas que tiene el direccionamiento indirecto.

2. ¿Se puede usar el direccionamiento indirecto para leer una tabla almacenada en la memoria de instrucciones? Explique.

3. Investigue en qué casos es mejor hacer una tabla en la memoria de datos y en que otros es preferibles hacerla en la memoria de instrucciones.

TEMA 5
DEFINICIÓN Y LECTURA DE TABLAS DE DATOS EN MEMORIA DE PROGRAMA

INTRODUCCIÓN

Existen en los MCUs PICs dos opciones para almacenar una tabla de datos. La primera consiste en usar la memoria de datos, donde los valores de cada uno de los elementos de la tabla —con la ayuda del direccionamiento indirecto— deben inicializarse por medio de múltiples instrucciones de tipo ***movlw*** y ***movwf***. La lectura de la tabla es posible usando tal modo direccionamiento y sus registros especiales. El segundo método permite definir la tabla en la memoria de programa usando ya sea la instrucción ***retlw*** o las directivas ***DT*** y ***DB***. La lectura de los elementos que conforman la tabla se logra ya sea con un "pseudo" direccionamiento indexado implementado con la escritura controlada del registro ***PCL*** o mediante las instrucciones y registros especiales para lectura de datos en memoria de instrucciones.

El experimento correspondiente a la definición y lectura de tablas de datos en memoria de programa tiene los siguientes objetivos prioritarios:

- Identificar los pasos necesarios para construir una tabla en la memoria de programa en el MCU PIC18F4550.
- Definir una tabla de datos por medio de la instrucción ***retlw***, con la directiva ***DT***.
- Analizar la característica de reescritura al registro ***PCL*** y diseñar con ella saltos controlados para leer cualquier elemento de una tabla en memoria de programa.
- Definir una tabla de datos usando la directiva ***DB*** y realizar su lectura por medio de la instrucción ***TBLRD***, el registro ***TABLAT*** y los registros apuntadores ***TBLPTRL***, ***TBLPTRH***, ***TBLPTRU***.

Los materiales y equipo necesarios para el correcto desarrollo del experimento 5 son los siguientes:

Cantidad	Descripción
1	Microcontrolador PIC18F4550
1	Microinterruptor (1P-1T).
1	*Display* de 7 segmentos de ánodo común.
2	Push-buttons.
8	Resistencias de 330 Ω.
8	Resistencias de 4.7 kΩ.
1	Diodo 1N4148
1	Tableta experimental

- Dispositivo programador compatible con PIC18F4550.
- Fuente de alimentación de CD.
- Probador lógico.
- Laboratorio equipado con computadoras que tengan instalado el MPLAB® X versión v5.0 o superior y software para dispositivo programador de PIC's.

Las actividades previas recomendadas son:

- Leer previamente todo el documento del experimento 5.
- Investigar la sintaxis de la instrucción ***retlw*** y las directivas ***DT*** y ***DB***. Asimismo, la instrucción ***TBLRD*** y los registros ***TABLAT***, ***TBLPTRL***, ***TBLPTRH*** y ***TBLPTRU***.
- Codificar en ensamblador en el programa propuesto apoyándose en los diagramas de flujo de las figuras 5.1-5.3.
- Llevar implementado en PROTEUS el circuito correspondiente.

EXPERIMENTO 5

DESARROLLO

1. Elaborar programa que muestre en un *display* 7 segmentos en el puerto B, en forma consecutiva, el ID de los integrantes del equipo. El programa responderá al dato que se introduce en el puerto "D" acorde a lo siguiente:

- Los bits 1 y 0 seleccionarán el nombre del integrante del equipo que se va a mostrar (3 integrantes). Si los bits están en 00, no se muestra nada.
- Los bits 7, 6 y 5 servirán para definir el tiempo que se van a mostrar cada letra (en segundos).
- El bit 0 del puerto "A" se presiona para comenzar a desplegar el nombre seleccionado.

Por ejemplo, si los integrantes del equipo son: Hugo (ID_1); Paco (ID_2) y Luis (ID_3), y se quiere mostrar el ID del integrante 2 (Paco) y además que cada digito se muestre durante 6 segundos, entonces en el puerto "D" deberá ponerse un 0xC2 ya que los bits del puerto deben estar en: "011x xx10", y los bits "no-importa" mostrados con x, se ponen como "0".

I. Tabla de datos con la instrucción *retlw*

Las tablas mensajes codifican los dígitos de los números de los IDs de los integrantes. La tabla usa como terminador el dato "0x00" para indicar al programa que el mensaje en cuestión terminó. A continuación, se muestra a manera de ejemplo la tabla para el ID "12345" (suponiendo display de ánodo común y orden de los segmentos 1 g e f d c b a).

```
TABLA1:   addwf   PCL, f, 0
          retlw   b'11111001'
          retlw   b'10100100'
          retlw   b'10011001'
          retlw   b'10010010'
          retlw   0x00
```

II. Tabla de datos con la directiva DT

Modifique su código para que ahora las tablas mensajes sean definidas con la directiva DT. Retomando el ejemplo anterior, la porción de código correspondiente sería ahora:

```
TABLA1:   addwf   PCL, f, 0
          dt      F9h, 0A4h, 99h, 92h, .0
```

Para ambos casos, asegúrese de organizar las tablas adecuadamente, para tal fin use la directiva ORG de una forma conveniente.

III. Tabla de datos con la directiva DB e instrucción TABLAT

Modifique su código para que ahora las tablas mensajes sean definidas con la directiva ***DB***. La lectura deberá realizarse con la directiva ***DB*** y realizar su lectura por medio de la instrucción ***TBLRD***, el registro ***TABLAT*** y los registros apuntadores ***TBLPTRL***, ***TBLPTRH***, ***TBLPTRU***.

```
        org     000020h
TABLA1: db      F9h, 0A4h, 99h, 92h, .0
```

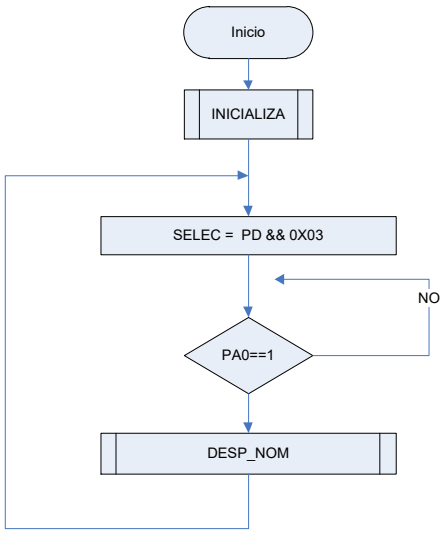


Figura 5.1: Diagrama de flujo del programa principal

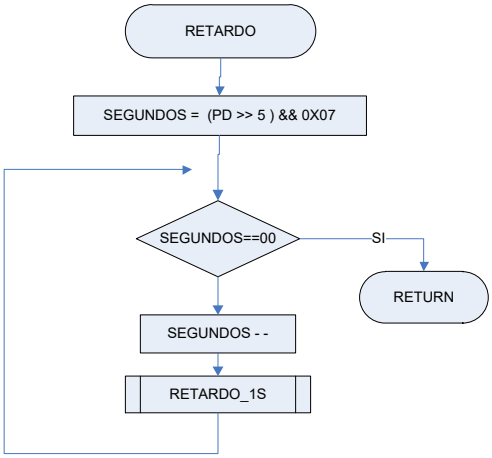


Figura 5.2: Diagrama de flujo de la subrutina RETARDO

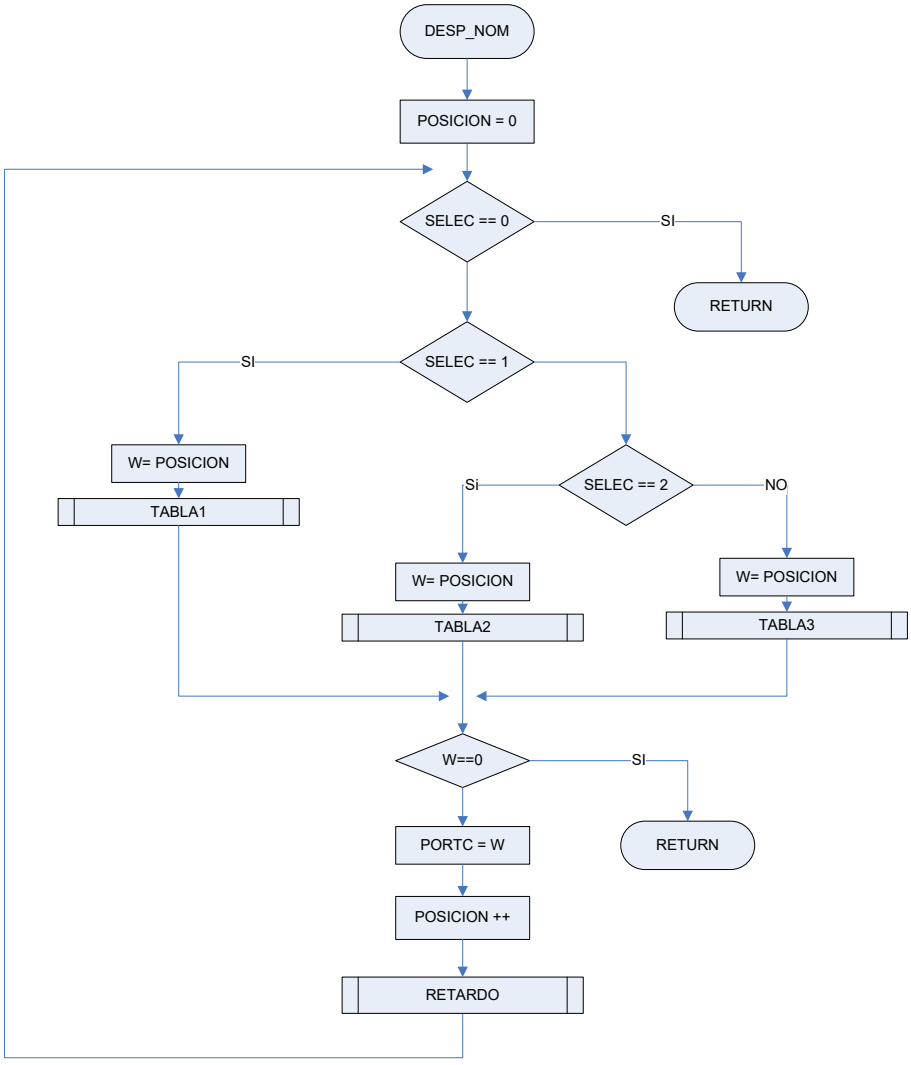


Figura 5.3: Diagrama de flujo de la subrutina DESP_NOM

ACTIVIDADES COMPLEMENTARIAS

1. Investigue algunas otras directivas del ensamblador que pueden ser usadas para el manejo de tablas en memoria de programa.
2. Consulte la hoja de datos y averigüe si dentro del conjunto de instrucciones del PIC18F4550, existen instrucciones/mnemónicos especiales para el manejo de tablas en la memoria *flash* de programa.
3. ¿Podrá un código ASM ejecutar un programa que automodifique la memoria de programa? Justifique su respuesta.

TEMA 6

DECODIFICACIÓN DE TECLADO MATRICIAL

INTRODUCCIÓN

Los principales elementos de entrada en un sistema basado en MCU son las teclas, pulsadores o interruptores. Sin embargo, su principal desventaja que requieren de una terminal E/S dedicada del MCU. Por tal motivo, cuando la cantidad de teclas es considerable, se debe explorar otras opciones. Un teclado matricial, tal como lo sugiere la figura 6.1, es un dispositivo de entrada compuesto de un arreglo de teclas dispuestas en una matriz filas-columnas (4x4 o 4x3) con la intención de reducir el número de líneas de entradas y salidas necesarias para controlarlo desde un microcontrolador. Proteus® cuenta en sus librerías soporte para teclado matricial tipo calculadora.

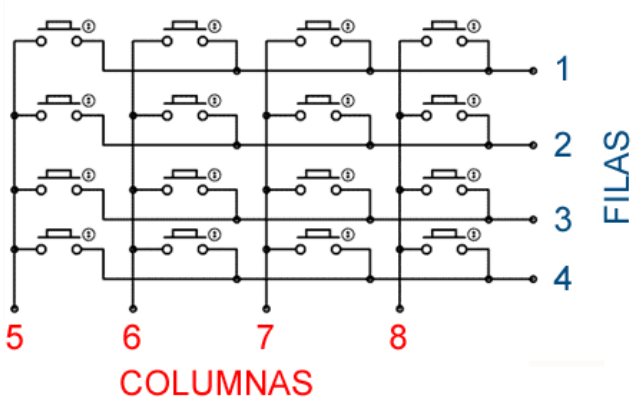


Figura 6.1: Diagrama eléctrico de un teclado matricial típico.

La mayoría de las veces, para controlar un teclado matricial, los puertos del MCU correspondientes a las filas se programan como salidas y los conectados a las columnas del teclado se programan como entradas. De tal forma que el objetivo principal del algoritmo para decodificar el teclado consiste en determinar la fila y columna que corresponde a la tecla que se presionó.

Lo anterior se logra rotando un valor lógico (ya sea 1 ó 0) en cada una de las líneas configuradas como salidas (filas en este caso) e inmediatamente después leer el estado lógico de las líneas conectadas como entrada (columnas).

Cuando el valor lógico que se rota es un 1 al algoritmo se le denomina ***walking ones*** y ***walking zeros*** cuando se trata de un 0. Otro algoritmo que se emplea para la decodificación de un teclado matricial es el de ***inversión de líneas***.

Con el experimento 6 se analizará el funcionamiento del teclado matricial y de los algoritmos para su decodificación, podrás reafirmar el conocimiento adquirido con respecto al direccionamiento indirecto, haciendo uso intensivo de él para diseñar e implementar un programa para la decodificación de un teclado matricial.

A continuación, se presenta una lista de equipo y material el cuál utilizarás durante el desarrollo de la actividad.

Cantidad	Descripción
1	Microcontrolador PIC18F4550
1	Teclado Matricial de 4x4
8 ó 1	Leds o una barra de leds
4	Resistencias de 330 Ω
8	Resistencias de 4.7 kΩ
1	*Display* de 7 segmentos de cátodo común
1	*Display* de 7 segmentos de ánodo común
1	Tableta experimental

- Dispositivo programador compatible con PIC18F4550.
- Fuente de alimentación de CD.
- Computadora que tenga instalado el MPLAB® X versión v5.0 o superior y software para dispositivo programador de PIC's.

Las actividades previas recomendadas son las siguientes:

- Leer previamente todo el documento del experimento 6.
- Construir en Proteus® cada uno de los circuitos planteados en la práctica.
- Llevar implementado en un *protoboard* el circuito de la figura 6.3.
- Investigar la distribución de pines del teclado matricial disponible en el laboratorio.

EXPERIMENTO 6

DESARROLLO

1. Edite en MPLAB IDE el programa mostrado en la figura 6.2. Escriba en las líneas en blanco, el valor correcto para desplegar los dígitos hexadecimales en el ***display*** de 7 segmentos de cátodo común. Considere también que la interconexión de este con el MCU es PA0 → a, PA1 → b,…, PA5→ f, y PA6 → g. Genere el proyecto de la forma acostumbrada, ensamble y depure hasta que no existan errores o ***warnings***.

```
; ************************************************************
; Sistemas Digitales III p2016
; Programa que realiza decodifica un teclado matricial 4x4
; con el algoritmo "walking ones" y con uso intensivo de
; direccionamiento indirecto
;
; Autor: ERA
; ************************************************************
      LIST        P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX       HEX

;************************************************************
;               BITS DE CONFIGURACION
;************************************************************
;********       Configuracion del Oscilador     **********
;       Osc interno, RA6 como pin, USB usa Osc EC
      CONFIG  FOSC    = INTOSCIO_EC

;********      Bits de configuración mas usados     **********
      CONFIG  PWRT    = ON    ; PWRT habilitado
      CONFIG  BOR     = OFF   ; Brown out deshabilitado
      CONFIG  WDT     = OFF   ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON    ; MCLR como entrada de reset
      CONFIG  PBADEN  = OFF   ; PB como entradas digitales
      CONFIG  LVP     = OFF   ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF   ; los bloques del codigo de
      CONFIG  CP1     = OFF   ; programa no protegidos
      CONFIG  CPB     = OFF   ; Boot sector no protegido
```

Figura 6.2: Listado con el programa para el teclado matricial.

```
; ***********************************************************
; Variables para la tabla de despliegue
; considerando display de cátodo común
      cblock      20h
      DIGITO_CERO
      DIGITO_UNO
      DIGITO_DOS
      DIGITO_TRES
      DIGITO_CUATRO
      DIGITO_CINCO
      DIGITO_SEIS
      DIGITO_SIETE
      DIGITO_OCHO
      DIGITO_NUEVE
      DIGITO_LETRA_A
      DIGITO_LETRA_B
      DIGITO_LETRA_C
      DIGITO_LETRA_D
      DIGITO_LETRA_E
      DIGITO_LETRA_F

; Variables para el programa ------------------------
      TEMP1
      TEMP2
      CONTADOR
      endc

;************************************************************
; Vector de Reset.
      ORG     0x0000
      goto    INICIO          ; Ve al inicio del programa pral

;************************************************************
; Vector de interrupcion
      ORG     0x0008
      goto    INICIO          ; Ve al inicio del programa pral

;************************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
      movlw   B'01100010'
      movwf   OSCCON          ; Oscilador interno a 4 MHz
      return

;************************************************************
; Subrutina que inicializa la tabla con los valores
; que se deben enviar al puerto A, Para desplegar
; el número correspondiente a la tecla presionada
CONF_TABLA
      lfsr    FSR0, DIGITO_CERO
; DIGITO_CERO
      movlw   _________
      movwf   POSTINC0, 0
```

Figura 6.2: Listado con el programa para el teclado matricial (continuación).

```
; DIGITO_UNO
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_DOS
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_TRES
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_CUATRO
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_CINCO
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_SEIS
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_SIETE
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_OCHO
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_NUEVE
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_LETRA_A
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_LETRA_B
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_LETRA_C
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_LETRA_D
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_LETRA_E
      movlw    _________
      movwf    POSTINC0, 0
; DIGITO_LETRA_F
      movlw    _________
      movwf    POSTINC0, 0
      return

;*****************************************************************
; Subrutina que configura PA como salida, el nibble
; bajo del PB como salida y el nibble alto del PB como entrada
CONF_PUERTOS
      clrf     LATA, 0         ;Limpia lacths del puerto A
      clrf     LATB, 0         ;Limpia lacths del puerto B
```

Figura 6.2: Listado con el programa para el teclado matricial (continuación).

```
      movlw   0Fh
      movwf   ADCON1, 0       ; Todos los pines como I/O dig
      clrf    TRISA, 0        ; Todos el PA como salidas
      movlw   0xF0            ; PB parte alta como entrada
      movwf   TRISB, 0        ; PB parte baja como salida
      return

;*****************************************************************
; Subrutina que rota a la izq. un "1" en la parte baja del PB
ROTA_LED
      movf  ..CONTADOR, w, 0
      rlncf ..WREG, w, 0
      addwf ..PCL, f, 0
      bra     F0
      bra     F1
      bra     F2
      bra     F3

F3    movlw   01h
      movwf   LATB            ; Escribe 0001 en puerto B
      lfsr    FSR0, 20h       ; Actualiza apuntador a la fila 0
      return

F2    movlw   02h
      movwf   LATB            ; Escribe 0010 en puerto B
      lfsr    FSR0, 24h       ; Actualiza apuntador a la fila 1
      return

F1    movlw   04h
      movwf   PORTB           ; Escribe 0100 en puerto B
      lfsr    FSR0, 28h       ; Actualiza apuntador a Fila 2
      return

F0    movlw   08h
      movwf   PORTB           ; Escribe 1000 en puerto B
      lfsr    FSR0, 2Ch       ; Actualiza apuntador a Fila 3
      return

;*****************************************************************
; Subrutina que decodifica la tecla presionada y manda a escribir en
; el display de cátodo común, el digito correspondiente

DECODIFICA
      swapf   PORTB, w, 0     ; Intercambia nibbles
      andlw   0Fh             ; Esmascara  nibble bajo
      movwf   TEMP1, 0        ; guarda columna en TEMP1
      btfss   TEMP1, 0, 0     ; Columna cero activa?
      bra     SIGUE0          ; No
      movlw   00h             ; Si
      bra     APUNTA          ; Checa las demás columnas
```

Figura 6.2: Listado con el programa para el teclado matricial (continuación).

```
SIGUE0
      btfss   TEMP1, 1, 0     ; Columna 1 activa?
      bra     SIGUE1          ; No
      movlw   01h             ; Si
      bra     APUNTA          ; Checa las demás columnas

SIGUE1
      btfss   TEMP1, 2, 0     ; Columna 2 activa?
      bra     SIGUE2          ; No
      movlw   02h             ; Si
      bra     APUNTA          ; Checa las demás columnas

SIGUE2
      btfss   TEMP1, 3, 0     ; Columna 3 activa?
      return                  ; No, no se presiono tecla
      movlw   03h             ; Si

APUNTA
      movf    PLUSW0, w, 0    ; Lee el contenido de la dir. del FSR + W
      movwf   PORTA, 0        ; y envialo al display
      return

;*****************************   Retardo 1ms      ********************
; Subrutina (bloqueante) que ejecuta un retardo de 1 ms por software
; decrementado el registro WREG 249 veces en un ciclo, asumiendo que
; la frecuencia de reloj es de un 1 MHz
;*****************************************************************
RETARDO_UN_MS
      movlw   .249
OTRO
      addlw   0xFF            ; Decrementa W
      btfss   STATUS, Z, 0    ; ¿Es igual a 0?
      bra     OTRO            ; No => seguimos esperando
      return                  ; Si => ya transcurrió 1 ms

;*****************************************************************
; Programa principal
INICIO
      rcall   CONF_TABLA
      rcall   CONF_PUERTOS
AQUI
      movlw   0xFF
      movwf   CONTADOR

OTRA_TECLA
      incf    CONTADOR, f, 0
      btfsc   CONTADOR, 2, 0
      bra     AQUI
      rcall   ROTA_LED
      rcall   RETARDO_UN_MS
      rcall   DECODIFICA
      bra     OTRA_TECLA

      END
```

Figura 6.2: Listado con el programa para el teclado matricial (continuación).

2. Elabore el diagrama de flujo del código mencionado en el punto anterior, incluyendo el de todas sus subrutinas.

3. Vea el listado del programa y escriba el nombre de todas las variables que se emplean en él.

_____

_____

4. Identifique las subrutinas y escriba que es lo que hace cada una de ellas (apóyese en los comentarios que tiene el programa).

Tabla 6.1. Subrutinas y su respectiva función para el manejo del teclado matricial.

Subrutinas	**Función**

5. Inserte un punto de ruptura (***BREAK-POINT***) en la instrucción ***rcall CONF_PUERTOS*** (Para insertar un punto de ruptura, dar doble *clic* con el ratón posicionado en la instrucción, para borrarlo dar doble *clic* de nuevo).

6. Ver el contenido de los **GPR's** con el menú ***View >> File Registers***, no olvide a seleccionar el modo de visualización ***Symbolic***. Con ***View >> Special Function Register***, se abre la ventana que nos permite ver los ***SFR***.

7. Borre el contenido de los ***GPR*** a través del menú ***Debugger >> Clear Memory >> GPR's.***

8. Simule un ***RESET*** al microcontrolador, con el botón , a continuación, ejecute la simulación del programa con el botón ***RUN*** .

9. Observe que cuando la simulación llega al punto de ruptura se detiene.

10. En las ventanas que muestran el contenido de los ***GPR's*** y ***SFR's*** se puede notar que el simulador nos indica con color rojo los registros que han cambiado. Anote el valor que tienen ahora las variables.

_____

_____

11. Ahora simule paso a paso la subrutina ***CONF_PUERTOS***. Explique brevemente que es lo que hace.

_____

_____

12. Continué ejecutando el programa hasta que llegar a la subrutina ***ROTA_LED***. ¿Qué valor tiene la variable ***CONTADOR***?

_____

_____

13. Ejecute paso a paso la subrutina ***ROTA_LED*** hasta la instrucción ***addwf PCL, f, 0*** ¿Cuál es su función dentro del programa?

_____

_____

14. En la rutina ***ROTA_LED*** hay 4 secciones de código cada una rotuladas con las etiquetas ***F0***, ***F1***, ***F2***, y ***F3***. ¿Cuál es la función que realizan?

_____

_____

15. Analice la subrutina ***DECODIFICA*** con la ayuda del simulador. Explique ¿Cómo se realiza la decodificación de la tecla pulsada?

_____

_____

16. Descargue su programa al microcontrolador e implemente el circuito de la figura 6.3. Verifique su funcionamiento.

17. Vuelva comentario la línea de código ***call RETARDO***, compile y reprograme el PIC de nuevo y experimente. ¿Funciona bien el teclado?

_____

Explique ¿por qué es necesaria esa línea de código?

_____

_____

18. Repita la práctica (simulación e implementación), modificando el código y circuito original para que ahora la decodificación se haga mediante la técnica ***walking zeros*** y considerando un ***display*** de ánodo común.

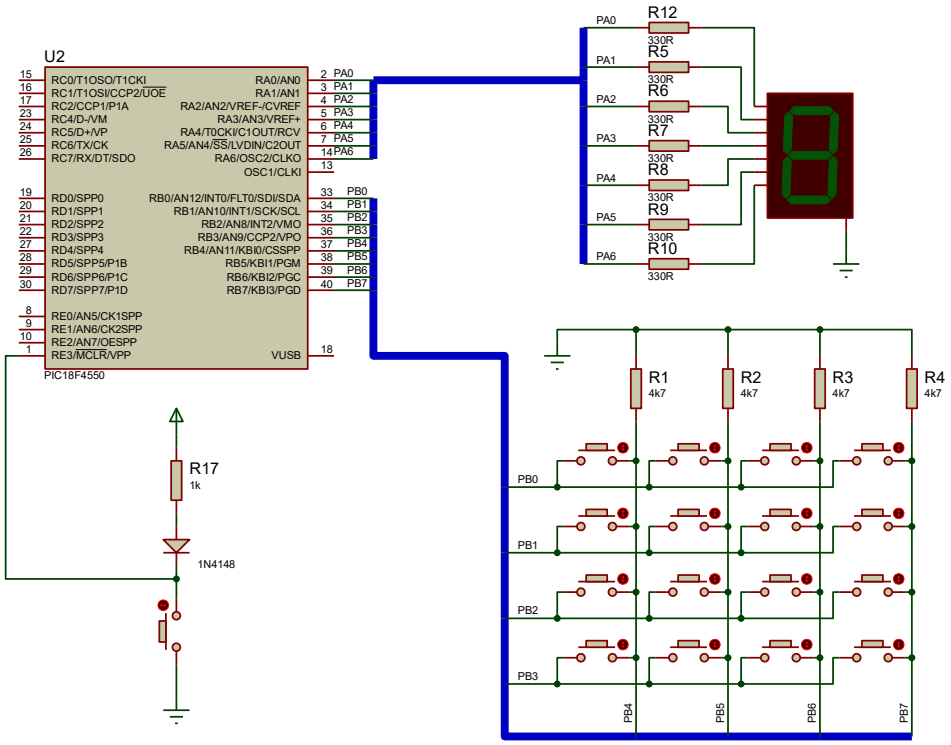


Figura 6.3: Circuito experimental para el teclado matricial.

ACTIVIDADES COMPLEMENTARIAS

1. ¿Será posible decodificar el teclado matricial mediante una tabla almacenada en la memoria de programa?

2. ¿Investigue si existe alguna configuración del teclado matricial de 4x4 que use menos líneas de E/S que uno convencional?

TEMA 7

INTERRUPCIONES EXTERNAS DEL PIC18F4550

INTRODUCCIÓN

Debido a que el PIC18F4550 es un microcontrolador de gama media alta, cuenta con un sistema de interrupciones muy completo que cubre a la mayoría de los periféricos de uso común en cualquier sistema *embedded*. Algunas de estas fuentes de interrupción son:

- Interrupciones externas.
- Interrupción por cambio de nivel lógico en el *nibble* alto del puerto B.
- Interrupciones por desborde de los temporizadores.
- Interrupciones por comparación exitosa en los temporizadores.
- Interrupciones de los comparadores analógicos.
- Interrupción por fin de conversión del ADC.
- Interrupciones de la parte transmisora del módulo USART mejorado.
- Interrupciones del receptor del módulo USART mejorado.
- Interrupciones del módulo CCP.
- Interrupciones por puerto USB.

Del listado anterior, solo las dos primeras se clasifican como interrupciones externas y las restantes como internas. Todas las interrupciones pueden enmascararse a través del bit GIE del registro INTCON (máscara global) y todas pueden ser enmascaradas en forma individual por medio de bits de control habilitadores locales (generalmente el nombre termina en IE).

Para cada una de las fuentes de interrupción existe un bit de bandera asociado que se activa cada vez que sucede un evento específico en cada uno de los módulos periféricos (los nombres de cada bit de bandera generalmente terminan con IF). La activación de cada una de estas banderas junto con la activación de su correspondiente máscara local y la máscara global (figura 7.1), son los que condicionan al PIC a ejecutar una rutina específica denominada ***rutina de servicio de interrupción (ISR).***

Cuando se produce una interrupción, el MCU termina de ejecutar la instrucción en curso, luego borra la máscara global ***GIE*** (para prevenir futuras interrupciones), almacena el contenido del registro PC en el ***stack*** y salta al vector de interrupción. Ahí el PIC ejecuta la instrucción almacenada en tal vector (regularmente una instrucción de salto) y procede a ejecutar la ISR. Finalmente, el MCU al encontrar la instrucción que señala el fin de la ISR, recupera del ***stack*** el valor del PC, desenmascara el bit GIE y el programa reanuda su ejecución en la siguiente instrucción.

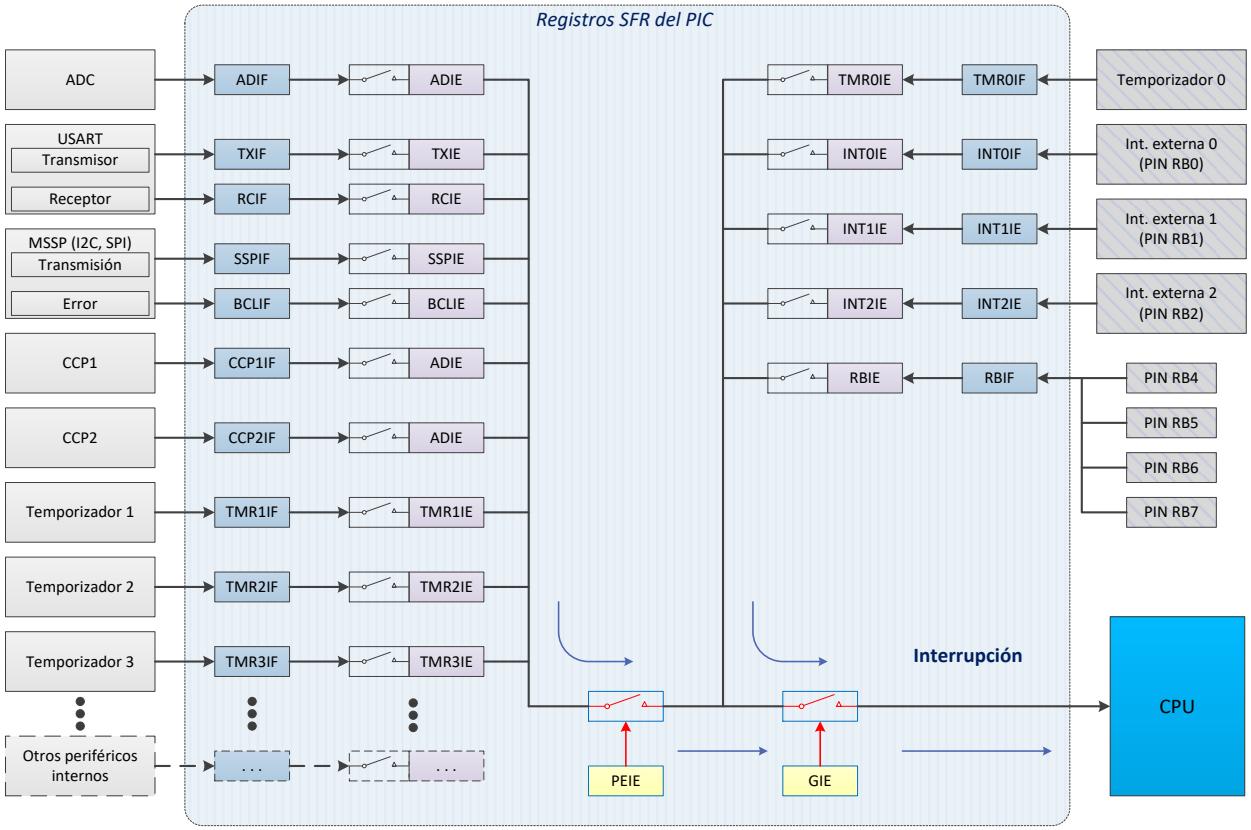


Figura 7.1 Diagrama simplificado del sistema de interrupciones en el PIC18F4550.

Todo este sofisticado sistema de interrupciones queda gobernado en su totalidad por medio de los siguientes 10 registros de control.

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1
- PIR2
- PIE1
- PIE2
- IPR1
- IPR2

Así mismo, cabe señalar que el sistema de interrupciones puede operar en modo compatibilidad con la familia PIC16FXXX y uno mejorado que permite establecer dos prioridades en las interrupciones y vectores de interrupciones configurables.

Por medio del siguiente experimento podrás analizar el sistema de interrupciones del PIC18F4550, identificar sus diferentes fuentes de interrupción, configurar adecuadamente los registros de control, banderas y bits de enmascaramiento relacionados con las interrupciones externas por las terminales RB0/INT0, RB1/INT1, y RB2/INT2 y las interrupciones por cambio de nivel en el *nibble* alto del puerto B.

De igual manera, aprenderás a configurar adecuadamente los registros de control para asignar prioridad a una interrupción y estructurar adecuadamente una rutina de servicio de interrupción (ISR) en lenguaje ensamblador.

El siguiente listado muestra el material y equipo necesario:

Cantidad	Descripción
1	Microcontrolador PIC18F4550
8	*Push-bottons*
8	Leds
8	Resistencias de 330 Ω
8	Resistencias de 4.7 kΩ
1	Resistencia de 10 kΩ
1	Diodo 1N4148
1	Tableta experimental

- Dispositivo programador compatible con PIC18F4550.
- Fuente de alimentación de CD.
- Computadora con MPLAB® X versión v5.0 o superior y software para dispositivo programador de PIC's.

Es recomendable leer previamente todo el documento del experimento 7, investigar los registros de configuración y bits relacionados al sistema de interrupciones y tener implementado en un *protoboard* el circuito de la figura 7.3 y 7.5

EXPERIMENTO 7

DESARROLLO

I. Interrupción externa por las terminales RB0/INT0, RB1/INT1 y RB2/INT2

Un simple suceso generado por un periférico externo puede ser detectado usando las terminales **RB0/INT, RB1/INT1 y RB2/INT2**. Siempre que se produzca una transición de señal en ellos, automáticamente se activará la bandera **INTxIF** correspondiente para registrar tal evento. Dichas interrupciones cuentan de forma individual con una máscara local **INTxIE** con el que se puede enmascarar o desenmascarar la interrupción en cuestión cuando sea necesario.

De igual manera, para seleccionar el flanco de la señal que provocará la interrupción en la terminal elegida, se debe configurar su bit **INTDEGx** en el registro **INTCON2**. De modo que, si el bit se pone a 1, la terminal va a ser sensible al flanco ascendente y si se pone en 0, al descendente.

1. Edite el programa mostrado en la figura 7.2 y ensamble en MPLAB hasta que no existan errores y *warnings*.

```
; *********************************************************
; Sistemas Digitales III p2016
; Programa que acepta una interrupcion externa por medio
; de la terminal RB0, mientras ejecuta un programa principal
;
; Fecha: 27/III/2020
; Autor: ERA
; *********************************************************
      LIST        P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX       HEX
;*********************************************************
;               BITS DE CONFIGURACION
;*********************************************************
;********       Configuracion del Oscilador     **********
;   Osc interno, RA6 como pin, USB usa Osc EC
      CONFIG  FOSC = INTOSCIO_EC
;********      Bits de configuración mas usados     *********
      CONFIG  PWRT    = ON    ; PWRT habilitado
      CONFIG  BOR     = OFF   ; Brown out deshabilitado
      CONFIG  WDT     = OFF   ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON    ; MCLR como entrada de reset
```

Figura 7.2: Programa para el circuito de la figura 7.3.

```
      CONFIG  PBADEN  = OFF   ; PB como entradas digitales
      CONFIG  LVP     = OFF   ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF   ; los bloques del codigo de
      CONFIG  CP1     = OFF   ; programa no protegidos
      CONFIG  CPB     = OFF   ; Boot sector no protegido

; ********************************************************
; Variables para el programa
      cblock      20h
      DIRECCION
      CONTADOR
      CANT_MS:2
      endc

;*********************************************************
; Vector de Reset.
      ORG         0x0000
      goto        INICIO            ; Ve al inicio del programa pral

;*********************************************************
; Vector de interrupcion
      ORG         0x0008
      goto        ISR_INT_EXT0      ; Ve a la rutina de servicio de int

;*********************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
      movlw       B'01100010'
      movwf       OSCCON            ; Oscilador interno a 4 MHz
      return

;*********************************************************
; Subrutina que configura PA y PD como salida digital, y el
; PB como entrada digital
CONF_PUERTOS
      clrf        LATA, 0           ; Limpia Lacths del puerto A
      clrf        LATB, 0           ; Limpia Lacths del puerto B
      clrf        LATD, 0           ; Limpia Lacths del puerto D
      movlw       0Fh
      movwf       ADCON1, 0         ; Todos los pines como I/O dig
      movlw       07h
      movwf       CMCON, 0          ; Apaga los comparadores
      clrf        TRISA, 0          ; Todo el PA como salidas
      clrf        TRISD, 0          ; Todo el PD como salidas
      movlw       0xFF
      movwf       TRISB, 0          ; Todo el PB como entrada
      bcf         INTCON2, RBPU, 0  ;Activa Rpull-up en todo PB
      bsf         LATD, 4, 0
      return
```

Figura 7.2: Programa para el circuito de la figura 7.3 (continuación).

```
;*************************************************************
; Subrutina que configura el pin RB0 como entrada de int. externa
CONF_INT_EXT0
      bcf         RCON, IPEN, 0     ; Interrupciones sin prioridad
      bsf         INTCON2, INTEDG0, 0 ;Configura flanco
      bcf         INTCON, INT0IF, 0 ; Limpia bandera de int. ext.
      bsf         INTCON, INT0IE, 0 ; Habilita int. ext. por RB0
      return

;*************************************************************
; Subrutina que desplaza cada 1/4 seg. un "1" de izquierda a
; derecha y viceversa en el PD
DESPLAZA_BIT_PD
      movf        DIRECCION, f, 0
      bz          DESP_LSB
      btfss       LATD, RD0, 0
      bra         NO_COND1
      bra         SI_COND1
NO_COND1
      rrncf       LATD, f, 0  ; Desplaza bit hacia el LSB
      return
SI_COND1
      clrf        DIRECCION, 0
      return

DESP_LSB
      btfss       LATD, RD7, 0
      bra         NO_COND2
      bra         SI_COND2
NO_COND2
      rlncf       LATD, f, 0  ; Desplaza bit hacia el MSB
      return
SI_COND2
      btg         DIRECCION, 0, 0
      return

;*************************************************************
; Rutina de servicio de interrrupcion (ISR) debido a INT0
; Incrementa el contador de interrupciones y lo despliega en
; el display de 7 segmentos.
ISR_INT_EXT0
      Incf        CONTADOR, f, 0    ; Contador++
      movlw       .10
      cpfseq      CONTADOR, 0       ; ¿Contador = 10?
      bra         NO_COND3          ; No
      clrf        CONTADOR, 0       ; Si, limpia contador
NO_COND3
      bcf         INTCON, INT0IF, 0; Limpia bandera de INT EXT0
      retfie
```

Figura 7.2: Programa para el circuito de la figura 7.3 (continuación).

```
; Subrutina que decodifica el valor de WREG a su correspondiente
; en 7 segmentos. Se asume display de cátodo común y la conexión
; de los segmentos con los pines puerto son
; bit 0-a, bit 1-b, ... , bit 6-g, bit 7-punto.
DECODIFICA
      addwf       PCL, f, .0
      retlw       b'00111111'
      retlw       b'00000110'
      retlw       b'01011011'
      retlw       b'01001111'
      retlw       b'01100110'
      retlw       b'01101101'
      retlw       b'01111101'
      retlw       b'00000111'
      retlw       b'01111111'
      retlw       b'01101111'

;*****************************************************************
; Subrutina (bloqueante) que genera un retardo de 250 ms aprox
; haciendo uso de la rutina RETARDO_VAR_MS
RETARDO_250_MS
      movlw       0x00              ;No => hacemos Cantms = 1000 1f4
      movwf       CANT_MS+1,.0
      movlw       0xFA
      movwf       CANT_MS,.0
      rcall       RETARDO_VAR_MS
      return

;*****************************************************************
; Subrutina RETARDO_VAR_MS
; Ejecuta un retardo de la cantidad de ms indicados mediante CANT_MS y
; CANT_MS+1, ambos son tratados como ; variables de 8 bits.
;*****************************************************************
RETARDO_VAR_MS
      rcall       RETARDO_UN_MS     ;realizamos un retardo de 1 ms
      decfsz      CANT_MS, F,.0     ;Cantms --, Cantms == 0
      bra         RETARDO_VAR_MS    ; no => vamos a
      movf        CANT_MS+1, W, .0
      btfsc       STATUS, Z         ; Cantms+1 == 0?
      return                        ; Si => retornamos Retardo_ms
      decf        CANT_MS+1, F, .0  ; No => decrementamos
      bra         RETARDO_VAR_MS    ; Continuamos con el ciclo
;***************************   Retardo 1ms      ********************
; Subrutina (bloqueante) que ejecuta un retardo de 1 ms por software
; decrementado el registro WREG 249 veces en un ciclo, asumiendo que
; la frecuencia de reloj es de un 1 MHz
;*****************************************************************
RETARDO_UN_MS
      movlw       .249
```

Figura 7.2: Programa para el circuito de la figura 7.3 (continuación).

```
OTRO
      addlw       0xFF              ; Decrementa W
      btfss       STATUS,Z,.0       ; ¿Es igual a 0?
      bra         OTRO              ; No => seguimos esperando
      return                        ; Si => ya transcurrió 1 ms

;*****************************************************************
; Programa principal
INICIO
      clrf        DIRECCION
      clrf        CONTADOR
      rcall       CONF_BASE_TIEMPO
      rcall       CONF_PUERTOS
      rcall       CONF_INT_EXT0
      bsf         INTCON, GIE, 0
AGAIN
      rcall       DESPLAZA_BIT_PD
      rcall       RETARDO_250_MS
      movf        CONTADOR, W, .0   ; Lee valor de pines del puerto A
      rlncf       WREG, W, .0
      rcall       DECODIFICA
      movwf       LATA, 0
      bra         AGAIN

      END
;*****************************************************************
```

Figura 7.2: Programa para el circuito de la figura 7.3 (continuación).

2. Implemente el circuito que se muestra en la figura 7.3

3. Descargue el programa al PIC18F4550, verificando que los bits de configuración queden programados para una frecuencia de oscilación interna a 4 MHz y con el bit ***MCLR*** habilitado.

4. Energice su circuito ¿Se observa algo?

_____

_____

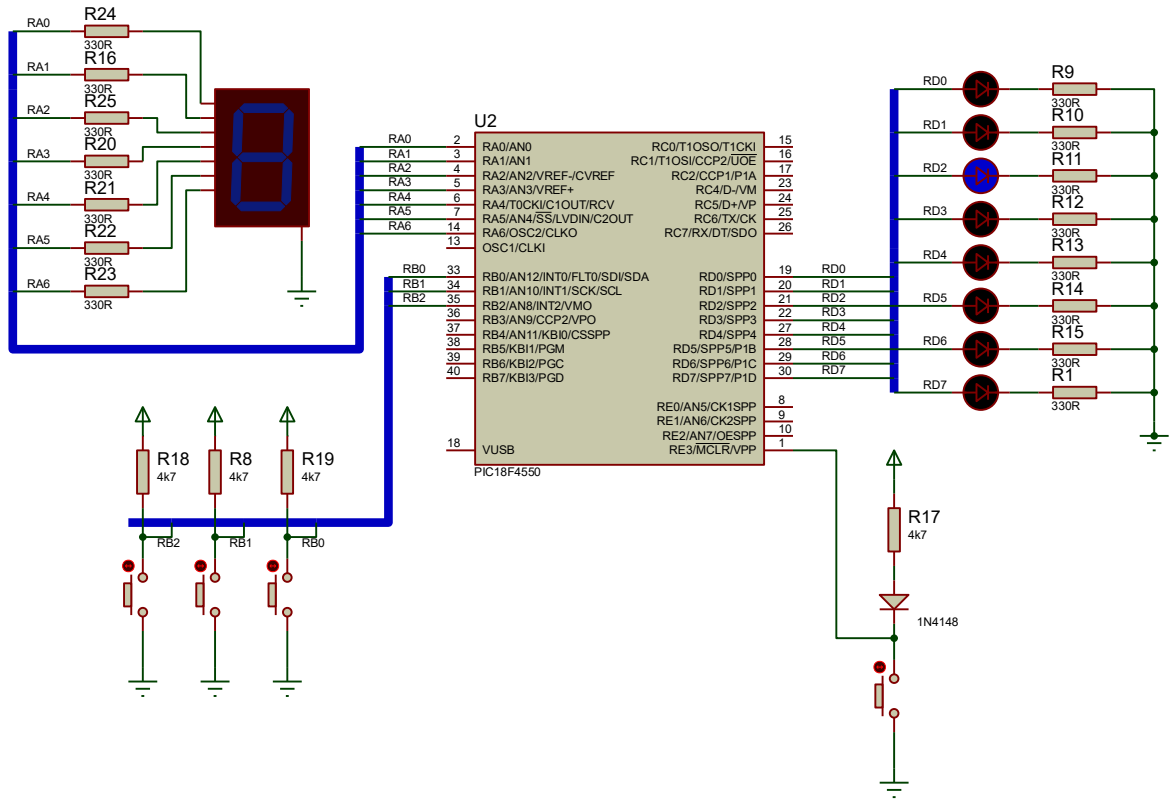


Figura 7.3: Circuito para el experimental para las interrupciones externas.

5. Oprima rápidamente el *push-button* de RB0 ¿Qué fue lo que pasó?

_____

_____

6. Oprima veces el *push-button*. Posteriormente actívelo manteniéndolo oprimido y luego suéltelo lentamente ¿Cuál es el comportamiento del circuito ahora?

_____

7. ¿Qué diferencias existe con respecto al punto 5?

_____

8. acuerdo a lo experimentado diga ¿En qué flanco se está disparando la interrupción, en el de subida o en el de bajada?

_____

9. Detenga la ejecución del programa pulsando el botón de *reset*.

10. Dentro de la subrutina ***CONF_INT_EXT0***, ubique la línea de código: ***bsf INTCON2, INTEDG0, 0*** y sustitúyala con ***bcf INTCON2, INTEDG0, 0***. Ensamble el programa y descárguelo en el MCU con la misma configuración de bits que se indicó en el punto 3.

11. Ejecute el programa y repita el punto 6. ¿Cómo se comporta ahora el circuito?

_____

_____

¿Con cuál flanco se está activando la interrupción

_____

12. Analizando la rutina de servicio de interrupción ***ISR_INT_EXT0*** puede notarse que la línea de código ***bcf INTCON, INT0IF, 0*** es fundamental para el funcionamiento correcto del programa, explique ¿por qué?

_____

_____

13. Modifique el programa con la finalidad de que por medio de interrupciones externas por INT2, al presionar el *push-button* conectado a esa terminal, el conteo del *display* disminuya en uno. Además, por medio de una interrupción por INT1, el contador se debe reiniciar a cero.

II. Interrupción por cambio de nivel lógico en la parte alta del puerto B.

El puerto B cuenta con una característica especial que nos permite programar una interrupción en su *nibble* alto, cada vez que el nivel lógico de cualquiera de las líneas cambia.

Los valores que se tienen en estas terminales se comparan constantemente con la última lectura realizada al puerto B y en el instante en que no coinciden, se activa el bit de bandera ***RBIF*** del registro **INTCON**. Para borrar ***RBIF***, es necesario primero leer el puerto B y posteriormente escribir un "0" en tal bit, **sin esta secuencia no se borrará, aunque se trate de escribir el "0" directamente.**

14. Edite el siguiente programa de la figura 7.4 en MPLAB y ensamble hasta que no existan errores ni *warnings*. La función del programa principal es desplazar constantemente un led en el puerto A, pero al mismo tiempo es capaz de procesar tanto en **interrupciones por cambio de nivel en la parte alta del puerto B y por flanco de subida en la entrada de interrupción externa 1** (terminal **RB1**).

```
; ********************************************************
; Sistemas Digitales III p2016
; Programa que acepta interrupciones (sin prioridad)
; por medio de la int. externa 1 y por cambio en RB
; mientras ejecuta un programa principal
;
; Fecha: 27/III/2020
; Autor: ERA
; ********************************************************
      LIST        P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX       HEX

;********************************************************
;               BITS DE CONFIGURACION
;********************************************************
;********       Configuracion del Oscilador     **********
;   Osc interno, RA6 como pin, USB usa Osc EC
    CONFIG  FOSC    = INTOSCIO_EC

;********      Bits de configuración mas usados     **********
    CONFIG  PWRT    = ON      ; PWRT habilitado
    CONFIG  BOR     = OFF     ; Brown out deshabilitado
    CONFIG  WDT     = OFF     ; Watchdog deshabilitado
    CONFIG  MCLRE   = ON      ; MCLR como entrada de reset
    CONFIG  PBADEN  = OFF     ; PB como entradas digitales
    CONFIG  LVP     = OFF     ; Prog bajo voltaje apagado
    CONFIG  DEBUG   = OFF
    CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
    CONFIG  CP0     = OFF     ; los bloques del codigo de
    CONFIG  CP1     = OFF     ; programa no protegidos
    CONFIG  CPB     = OFF     ; Boot sector no protegido

; ********************************************************
; Variables para el programa
      cblock      20h
      DIRECCION
      CANT_MS:2
      AUX
      INDICE
      endc

;*************************************************************
; Vector de Reset.
      ORG         0x0000
      goto        INICIO            ; Ve al inicio del programa pral
```

Figura 7.4: Programa para el circuito experimental de interrupciones de puerto B y externas

```
;*****************************************************************
; Vector de interrupcion
      ORG         0x0008
      goto        ISR_INTERRUPCIONES ; Ve a la ISR

;*****************************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
      movlw       B'01100010'
      movwf       OSCCON            ; Oscilador interno a 4 MHz
      return

;*****************************************************************
; Subrutina que configura PA y PD como salida digital, y el
; PB como entrada digital
CONF_PUERTOS
      Clrf        LATA, 0           ; Limpia Lacths del puerto A
      clrf        LATB, 0           ; Limpia Lacths del puerto B
      clrf        LATD, 0           ; Limpia Lacths del puerto D
      movlw       0Fh
      movwf       ADCON1, 0         ; Todos los pines como I/O dig
      movlw       07h
      movwf       CMCON, 0          ; Apaga los comparadores
      clrf        TRISA, 0          ; Todo el PA como salidas
      clrf        TRISD, 0          ; Todo el PD como salidas
      movlw       0xFF
      movwf       TRISB, 0          ; Todo el PB como entrada
      bcf         INTCON2, RBPU, 0  ; Activa Rpull-up en todo PB
      bsf         LATD, 4, 0
      return

;*****************************************************************
; Subrutina que configura las interrupciones por pin RB0 y
; por cambio en nibble alto de PB (RB4-RB7)
CONF_INTERRUPCIONES
      bcf         RCON, IPEN, 0           ; Interrupciones sin prioridad

; Configura interrucpiones por cambio en RB7-RB4
      movff       PORTB, AUX              ; Elimina mismatch en PB
      bcf         INTCON, RBIF            ; Borra bandera por cambio en PB
      bsf         INTCON, RBIE            ; Habilita int. por cambio en PB

; Configura interrupción externa 1
      Bsf         INTCON2, INTEDG1, 0     ; Configura flanco
      bcf         INTCON3, INT1IF, 0      ; Limpia bandera de int. ext.
      bsf         INTCON3, INT1IE, 0      ; Habilita int. ext.1 (RB1)
      return
```

Figura 7.4: Programa para el circuito experimental de interrupciones de puerto B y externas (continuación).

```
;*****************************************************************
; Subrutina que desplaza cada 1/4 seg. un "1" de izquierda a
; derecha y viceversa en el PD
DESPLAZA_BIT_PD
      movf        DIRECCION, f, 0
      bz          DESP_LSB
      btfss LATD, RD0, 0
      bra         NO_COND1
      bra         SI_COND1
NO_COND1
      rrncf       LATD, f, 0              ; Desplaza bit hacia el LSB
      return
SI_COND1
      clrf        DIRECCION, 0
      return
DESP_LSB
      btfss LATD, RD7, 0
      bra         NO_COND2
      bra         SI_COND2
NO_COND2
      rlncf LATD, f, 0              ; Desplaza bit hacia el MSB
      return
SI_COND2
      btg         DIRECCION, 0, 0
      return

;*****************************************************************
; Rutina de servicio de interrrupcion (ISR)
; Por ser un vector de interrupcion único, primero identifica la
; fuente de interrupcion (INT. EXT1 o RB) y con base en esto ejecuta la
; tarea asignada a cada una
ISR_INTERRUPCIONES
      movlw       .6
      movwf       INDICE, 0
CICLO_H
      movlw       76h
      movwf       LATA, 0                 ; Despliega "H" (High priority)
      rcall       RETARDO_300_MS          ; en el display
      movlw       00h
      movwf       LATA, 0
      rcall       RETARDO_300_MS
      decfsz      INDICE, f, 0
      bra         CICLO_H

; Identifica la fuente de interrupcion
      btfss       INTCON3, INT1IF, 0
      bra         SERVICIO_RB
SERVICIO_INT_EXT1
      movlw       .5
      movwf       INDICE, 0
```

Figura 7.4: Programa para el circuito experimental de interrupciones de puerto B y externas (continuación).

```
CICLO_IE1
      Movlw       30h               ; Despliega "I"
      movwf       LATA, 0           ; en el display
      rcall       RETARDO_300_MS    ; Despliega "E"
      movlw       79h               ; en el display
      movwf       LATA, 0
      rcall       RETARDO_300_MS
      movlw       00h
      movwf       LATA, 0
      rcall       RETARDO_300_MS
      decfsz      INDICE, f, 0
      bra         CICLO_IE1
      bcf         INTCON3, INT1IF, 0      ; Borra bandera de INT EXT1

SERVICIO_RB
      btfss       INTCON, RBIF, 0
      bra         SALIR_ISR
      movff       PORTB, AUX
      movlw       .5
      movwf       INDICE, 0
CICLO_RB
      Movlw       50h               ; Despliega "r"
      movwf       LATA, 0           ; en el display
      rcall       RETARDO_300_MS
      movlw       7Ch               ; Despliega "b"
      movwf       LATA, 0           ; en el display
      rcall       RETARDO_300_MS
      movlw       00h
      movwf       LATA, 0
      rcall       RETARDO_300_MS
      decfsz      INDICE
      bra         CICLO_RB
      bcf         INTCON, RBIF, 0   ; Borra bandera de RB
SALIR_ISR
      retfie      1

;*********************************************************************
; Subrutina (bloqueante) que genera un retardo de 300 ms aprox
; haciendo uso de la rutina RETARDO_VAR_MS
RETARDO_300_MS
      movlw       0x01              ; Cant_ms = 300 (12Ch)
      movwf       CANT_MS+1,.0
      movlw       0x2C
      movwf       CANT_MS,.0
      rcall       RETARDO_VAR_MS
      return
```

Figura 7.4: Programa para el circuito experimental de interrupciones de puerto B y externas (continuación).

```
;*********************************************************************
; Subrutina (bloqueante) que genera un retardo de 250 ms aprox
; haciendo uso de la rutina RETARDO_VAR_MS
RETARDO_250_MS
      movlw       0x00              ; No => hacemos Cantms = 1000 1f4
      movwf       CANT_MS+1,.0
      movlw       0xFA
      movwf       CANT_MS,.0
      rcall       RETARDO_VAR_MS
      return

;*********************************************************************
; Subrutina RETARDO_VAR_MS
; Ejecuta un retardo de la cantidad de ms indicados
; mediante CANT_MS y CANT_MS+1, ambos son tratados como
; variables de 8 bits.
;*********************************************************************
RETARDO_VAR_MS
      rcall       RETARDO_UN_MS     ; realizamos un retardo de 1 ms
      decfsz      CANT_MS, F,.0     ; Cantms --, Cantms == 0
      bra         RETARDO_VAR_MS    ; no => vamos a
      movf        CANT_MS+1, W, .0
      btfsc       STATUS, Z         ; Cantms+1 == 0?
      Return                        ; Si => retornamos Retardo_ms
      decf        CANT_MS+1, F, .0  ; No => decrementamos
      bra         RETARDO_VAR_MS    ; Continuamos con el ciclo

;*************************   Retardo 1ms   ****************************
; Subrutina (bloqueante) que ejecuta un retardo de 1 ms por software
; decrementado el registro WREG 249 veces en un ciclo, asumiendo que la
; frecuencia de reloj es de un 1 MHz
;*********************************************************************
RETARDO_UN_MS
      movlw       .249
OTRO
      addlw       0xFF              ; Decrementa W
      btfss       STATUS,Z,.0       ; ¿Es igual a 0?
      bra         OTRO              ; No => seguimos esperando
      return                        ; Si => ya transcurrió 1 ms

;*****************************************************************
; Programa principal
INICIO
      clrf        DIRECCION, 0
      movff       PORTB, AUX
      rcall       CONF_BASE_TIEMPO
      rcall       CONF_PUERTOS
      rcall       CONF_INTERRUPCIONES
      bsf         INTCON, GIE, 0
```

Figura 7.4: Programa para el circuito experimental de interrupciones de puerto B y externas (continuación)

```
AGAIN
      rcall       DESPLAZA_BIT_PD
      rcall       RETARDO_250_MS
      bra         AGAIN

      END
```

Figura 7.4: Programa para el circuito experimental de interrupciones de puerto B y externas (continuación).

15. Modifique el circuito de la figura 7.3 en la siguiente forma (figura 7.5): Desconecte los botones conectados a RB0 y RB2 conéctelos a las terminales RB6 y RB7. Estas terminales pueden activar interrupciones por cambio de nivel. Encienda el circuito y describa lo que observa en los leds del circuito.

_____

_____

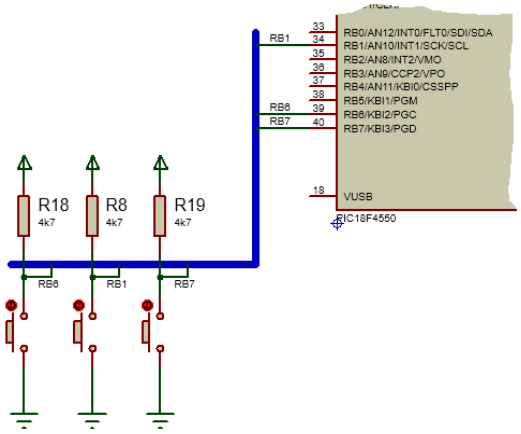


Figura 7.5. Modificación propuesta para el circuito inicial.

16. ¿Qué parte del programa se está ejecutando para producir este comportamiento?

_____

17. Del listado de la figura 7.4 en la rutina de servicio a interrupción ***ISR_INTERRUPCIONES***, ¿qué función tienen las líneas de código englobadas entre la etiqueta ***CICLO_H*** y y la instrucción ***bra CICLO_H***?

_____

18. Presione **sin soltar** el *push-button* conectado a RB6, observe lo que ocurre con los leds y el *dis play*. Espere a que los leds vuelvan a comportarse como antes de presionar el botón. Libere el *push-button* y explique ahora lo que ocurre con los leds y el *display*.

_____

_____

19. Repita el paso anterior pero ahora presionando y soltando RB7. ¿Existe alguna diferencia en el comportamiento?

_____

20. Dentro de la rutina ***ISR_INTERRUPCIONES*** explique ¿qué hace la porción de código que procesa el llamado de la interrupción externa 1 (RB1)?

_____

_____

21. Dentro de la rutina ***ISR_INTERRUPCIONES*** explique ¿qué hace la porción de código que procesa el llamado de la interrupción por cambio de nivel en la parte alta del puerto B?

_____

_____

22. Observe que la "interrupción por cambio en RB" se activa cuando se cambia el nivel de RB6 o se cambia el nivel RB7. Por otra parte, observe que la variable ***aux*** tiene la función de almacenar el valor del puerto B en cada interrupción "por cambio de RB", por lo que nos puede servir para identificar cuál bit ha cambiado desde la última lectura.

Pruebe el funcionamiento del circuito y explique ¿qué ocurre cuando hay un cambio en RB7?, ¿qué ocurre cuando hay un cambio en RB6? y explique las diferencias.

_____

_____

23. Modifique el código para que ahora la ISR también procese peticiones por cambio en RB6. Ésta deberá ejecutarse cuando se detecte un flanco descendente en esta terminal. La tarea de la rutina será mostrar en el *display* el mensaje ***rb6***.

Escriba a continuación el código que debe incluirse en el programa para lograr esto.

_____

_____

_____

_____

_____

_____

_____

_____

ACTIVIDADES COMPLEMENTARIAS

Investigue los siguientes cuestionamientos.

1. ¿Cuántos niveles de anidamiento son permitidos por interrupción', ¿de qué depende esto último

2. ¿Cuáles son las diferencias en la "interrupción externa" entre el PIC16F887 y el PIC18F4550?

3. ¿Cuáles son las diferencias en la "interrupción por cambio del PB" entre el PIC16F887 y el PIC18F4550?

4. ¿Existe alguna razón por la cual la interrupción por cambio del puerto B se active sólo en el *nibble* alto del dicho puerto?

TEMA 8
LOS TEMPORIZADORES PROGRAMABLES DEL PIC18F4550

INTRODUCCIÓN

Los módulos temporizadores son uno de los principales periféricos con los que puede tener en su haber los microcontroladores, ya que permiten la medición de tiempo, el monitoreo de frecuencia, la generación de eventos periódicos y el control preciso de eventos. El microcontrolador PIC18F4550 tiene implementado un sofisticado módulo de temporizadores conformado por:

- Timer0.
- Timer1.
- Timer2.
- Timer3.

De los tres, el Timer0 es el más empleado de forma individual, ya que los tres últimos suelen ser usados como periférico de apoyo para el módulo CCP (Capture/Compare/PWM).

Todos estos temporizadores pueden configurarse de manera flexible para adaptarse a diferentes requerimientos del sistema. Tienen la capacidad de generar interrupciones en intervalos predefinidos o midiendo eventos externos, con la opción de operar en modo de contador o temporizador. Además, pueden utilizar diversas fuentes de reloj, desde el oscilador interno hasta señales externas, lo que brinda versatilidad en el diseño de sistemas embebidos.

En este experimento, se explora la arquitectura y configuración de los temporizadores 0 y 1 del PIC18F4550, analizando su funcionamiento interno, los registros clave involucrados. Abordando aspectos esenciales para la programación en ensamblador y con ejemplos prácticos para ilustrar su aplicación.

Mediante el experimento 8, se desea lograr los siguientes objetivos:

- Identificar los diferentes temporizadores disponibles en el MCU PIC18F4550, así como sus registros de control que condicionan su funcionamiento.
- Comprender los conceptos de ***preescalador*** y ***pos-escalador*** en el contexto de los temporizadores de los MCU's PIC.
- Configurar en lenguaje ensamblador (ASM) los registros de control asociados al ***Timer0*** para que funcione en modo temporizador de 8 bits.
- Configurar en lenguaje ensamblador los registros de control del ***Timer1*** para su funcionamiento en modo contador.

El material y equipo que requeriremos se lista a continuación:

Cantidad	Descripción
1	Microcontrolador PIC18F4550
1	*Push-botton*
16 ó 2	Leds o barra de leds
16	Resistencias de 330 Ω
1	Resistencia de 10 kΩ
1	Diodo 1N4148
1	Tableta experimental

- Dispositivo programador compatible con PIC18F4550.
- Fuente de alimentación de CD.
- Osciloscopio.
- Generador de funciones.
- Laboratorio equipado con computadoras que tengan instalado el MPLAB® X versión v5.0 o superior y software para dispositivo programador de PIC's.

Para lograr los objetivos del experimento 8, se recomienda las siguientes acciones previas al su desarrollo:

- Leer previamente todo el documento del experimento 8.
- Llevar implementado en un *protoboard* el circuito de la figura 8.2 y 8.4
- Investigar en la hoja de datos del PIC18F4550, los diagramas a bloques de la arquitectura interna del Timer0, Timer1 y Timer3.

EXPERIMENTO 8

DESARROLLO

I. El Timer0 en modo temporizador

El Timer0, como lo muestra la figura 8.1, es un temporizador/contador que consta de un registro de lectura/escritura que puede configurarse (bit T08BIT) para un conteo de 8 ($TMR0_L$) o 16 bits ($TMR0_H$:$TMR0_L$). Puede ser programado (con el bit T0CS) para incrementar su conteo ya sea con impulsos externos en la terminal RA4/T0CKI (modo contador) o por medio de la señal interna $F_{osc}/4$ (modo temporizador). De igual forma, con la ayuda del bit T0SE es posible configurar el flanco con el que el conteo se incrementará en el caso que la señal sea externa. Asimismo, en modo 8 bits cuando el conteo del $TMR0_L$ pasa del valor FFh al 00h ó de FFFFh a 0000h en modo 16 bits, se activa la bandera TMR0IF en el registro INTCON. Con ella es posible programar una interrupción cada vez que se presente este evento por medio del bit TMR0IE.

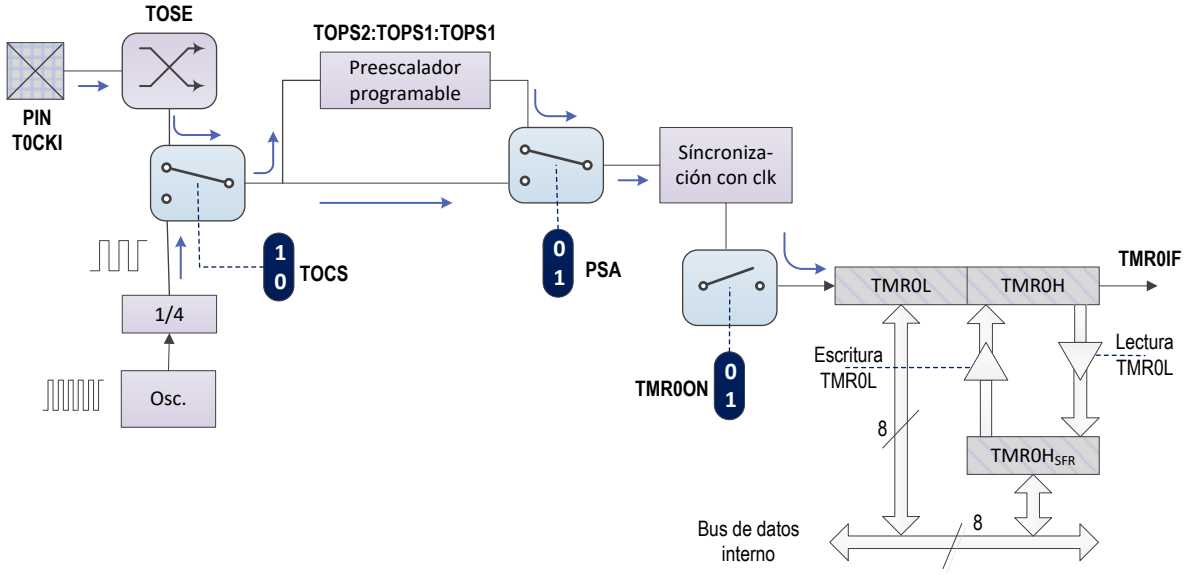


Figura 8.1: Arquitectura interna del módulo temporizador 0 (timer 0)

Cabe señalar que cada vez que se ejecuta una instrucción de escritura que modifica el valor en el registro $TMR0_L$, se producirá un retardo de dos ciclos de instrucción durante los cuales se inhibe tanto el predivisor (*prescaler*) como el $TMR0_L$. Será necesario tener en cuenta esa inhibición temporal a la hora de realizar una precarga (compensar sumando los ciclos de instrucción que "se pierden").

1. Edite en MPLAB el programa fuente en ensamblador que se muestra en la figura 8.2.

```
; *********************************************************
; Sistemas Digitales III p2016
; Programa que realiza un "blinking" del PB cada 1 segundo
; empleando el timer0 en modo temporizador de 8 bits.
; Fecha: 27/IV/2020
; Autor: ERA
; *********************************************************
      LIST        P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX       HEX

;*********************************************************
;               BITS DE CONFIGURACION
;*********************************************************
;********       Configuracion del Oscilador     **********
;   Osc interno, RA6 como pin, USB usa Osc EC
      CONFIG  FOSC    = INTOSCIO_EC

;********      Bits de configuración mas usados     *********
      CONFIG  PWRT    = ON    ; PWRT habilitado
      CONFIG  BOR     = OFF   ; Brown out deshabilitado
      CONFIG  WDT     = OFF   ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON    ; MCLR como entrada de reset
      CONFIG  PBADEN  = OFF   ; PB como entradas digitales
      CONFIG  LVP     = OFF   ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF   ; los bloques del codigo de
      CONFIG  CP1     = OFF   ; programa no protegidos
      CONFIG  CPB     = OFF   ; Boot sector no protegido

; *********************************************************
; Variables para el programa
      cblock      20h
      CONT_DESB
      endc

;*********************************************************
; Vector de Reset.
      ORG         0x0000
      goto        INICIO          ; Ve al inicio del programa pral

;*********************************************************
; Vector de interrupcion de alta prioridad
      ORG         0x0008
      goto        ISR_INT_TIMER0    ; Ve a la ISR
```

Figura 8.2: Programa en lenguaje ASM para el experimento con el temporizador 0.

```
;*************************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
      movlw       B'01100010'
      movwf       OSCCON            ; Oscilador interno a 8 MHz
      return

;*************************************************************
; Subrutina que configura PB y PD como salida digital
CONF_PUERTOS
      clrf        LATB, 0           ; Limpia Lacths del puerto B
      movlw       0Fh
      movwf       ADCON1, 0         ; Todos los pines como I/O dig
      clrf        TRISB, 0          ; Todo el PB como salidas
      return

;*************************************************************
; Subrutina que configura el Timer0 en modo temporizador de 8 bits
CONF_TIMER0
      movlw       0xF7              ; Configura la precarga
      movwf       TMR0L             ; del timer0
      bsf         T0CON, T08BIT, 0  ; Timer0 en modo 8 bits
      bcf         T0CON, T0CS, 0    ; Timer0 en modo temporizador
      bcf         T0CON, PSA, 0     ; Preescalador habilitado
      bsf         T0CON, T0PS0, 0   ; Preescalador 1:______
      bcf         T0CON, T0PS1, 0
      bsf         T0CON, T0PS2, 0
      bcf         INTCON, TMR0IF, 0 ; Borra la bandera del timer0
      bsf         INTCON, TMR0IE, 0 ; Desenmascara bit del timer0
      bsf         T0CON, TMR0ON, 0  ; Timer0 encendido
      return

;*************************************************************
; Rutina de servicio de interrrupcion (ISR), se activa en cada
; desborde del timer0. Son 62 desbordes en total

ISR_INT_TIMER0
      decfsz      CONT_DESB, f, 0   ; Decrementa desbordes
      bra         AUN_NO            ; Desbordes = 0? NO
      comf        LATB, f, 0        ; Complementa PB
      movlw       .62               ; Inicializa la cantidad
      movwf       CONT_DESB, 0      ; de desbordes
      movlw       0xF7              ; Configura la precarga
      movwf       TMR0L, 0          ; del timer0
AUN_NO
      bcf         INTCON, TMR0IF, 0 ; Borra bandera del timer0
      retfie
```

Figura 8.2: Programa en lenguaje ASM para el experimento con el temporizador 0 (cont.)

```
;*****************************************************************
; Programa principal
INICIO
      movlw       .62                     ; Inicializa la cantidad
      movwf       CONT_DESB, 0            ; de desbordes
      rcall       CONF_BASE_TIEMPO
      rcall       CONF_PUERTOS
      rcall       CONF_TIMER0
      bsf         INTCON, GIE, 0          ; Activa mascara global
      bra         $                       ; Ciclo ocioso

      END
```

Figura 8.2: Programa en lenguaje ASM para el experimento con el temporizador 0 (cont.)

2. Elabore el proyecto correspondiente en MPLAB y proceda a ensamblar el programa en la forma acostumbrada.

3. Implemente el circuito de la figura 8.3, descargue el archivo HEX al PIC y ejecútelo. Verifique el funcionamiento del circuito viendo el comportamiento de los leds del puerto B.

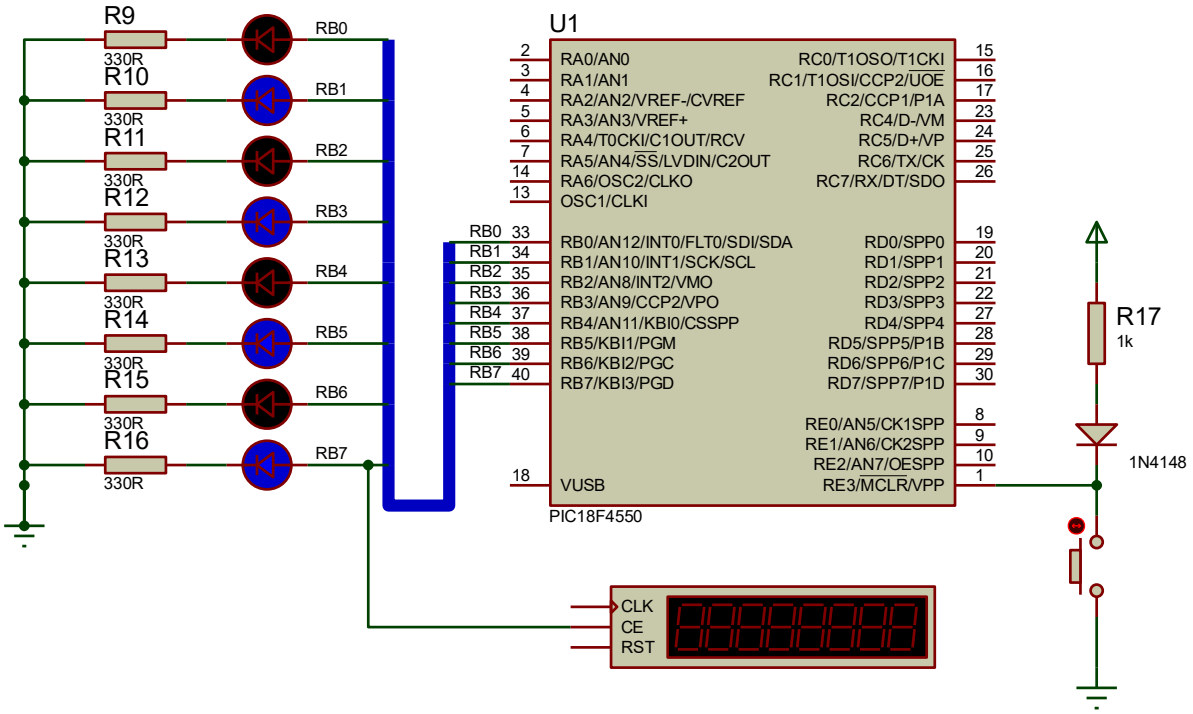


Figura 8.3: Circuito de experimentación para el Timer0.

4. Observe que el programa principal está formado por 7 líneas de código y básicamente lo que hace es configurar la velocidad del reloj interno, el puerto B y el Timer0. Asimismo, se inicializa la variable ***cont_desb*** y permanecer en un ciclo ocioso. Cada vez que se desborda el registro

TMR0L, el PIC generará una interrupción debido a este evento y realizará las tareas programadas en la rutina de servicio de interrupción, las cuales en este caso son decrementar el contenido de la variable ***cont_desbordes*** y cuando sea cero, se complementará lo desplegado en el puerto B.

5. Del archivo fuente anterior, dentro de la subrutina ***CONF_TIMER0***, localice las líneas de código:

```
bsf  T0CON, T0PS0, 0
bcf  T0CON, T0PS1, 0
bsf  T0CON, T0PS2, 0
```

sustitúyalas ahora por las siguientes:

```
bcf  T0CON, T0PS0, 0
bcf  T0CON, T0PS1, 0
bcf  T0CON, T0PS2, 0
```

Almacene el archivo, ensámblelo nuevamente, descargue el programa al PIC y ejecútelo. ¿Cuál es el comportamiento de los leds ahora?

_____

_____

Exactamente ¿qué fue lo que hizo la nueva línea de código?

_____

¿De qué manera influye en el programa?

_____

¿Cuál es el tiempo que permanecerán encendidos los leds con este cambio?

_____

Escriba la fórmula mediante la cual es posible calcular el tiempo que dura el retardo.

_____

6. Analizando la rutina de servicio de interrupción ***ISR_INT_TIMER0***, puede notarse que la función de la penúltima línea de código es fundamental para el buen funcionamiento del programa. Explique: ¿Por qué?

_____

7. Verifique su respuesta eliminando del programa la línea de código a la que hace referencia el punto anterior, ensamble el proyecto, reprograme el PIC y ejecútelo. ¿Qué es lo que sucede ahora?

_____

_____

8. En la subrutina CONF_TIMER0, ubique la línea de código:

```
bsf  T0CON, T08BIT, 0
```

modifíquela con:

```
bcf  T0CON, T08BIT, 0
```

Almacene el archivo, proceda a su ensamble, descargue el programa al PIC y ejecútelo. ¿Cuál es el comportamiento de los leds comparado con el paso anterior?

_____

_____

¿qué fue lo que hizo la nueva línea de código en el funcionamiento de Timer0?

_____

¿Cuál es el tiempo que permanecerán encendidos los leds con este cambio?

_____

Realice los cálculos para predecir el retardo de tiempo mediante la fórmula y compárelo con el obtenido en el circuito.

_____

9. Modifique el programa para que el Timer0 en modo de 16 bits sea capaz de generar un retardo de 1 segundo. Verifique su funcionamiento en el circuito de la figura 8.1.

II. El Timer1 en modo contador

El Timer1 es un módulo temporizador/contador ascendente de 16 bits que puede ser escrito o leído en cualquier instante, está implementado en dos registros específicos $TMR1_H$ y $TMR1_L$, que contienen el valor de las cuentas a cada momento (figura 8.4). El valor del registro $TMR1_H$:$TMR1_L$ incrementa desde 0000h hasta FFFFh, el cual al reanudar la cuenta activa el señalizador TMR1IF.

Como fuente de impulsos de reloj existen tres alternativas:

- Generación interna (4*Tosc).
- Generación mediante un oscilador externo controlado por cristal que se conecta a las terminales RC0/T1OSO/T1CKI y RC1/T1OSI/CCP2. El oscilador se activa poniendo a 1 el bit T1OSCEN del registro T1CON. El bit del TMR1CS del registro T1CON selecciona entre reloj interno y externo.
- Trabajar en modo contador de eventos, cuando los impulsos externos a contar se introducen al pin RC0/T1OSO/T1CKI. La fuente de los impulsos de reloj se aplica al preescalador que los divide por 1, 2, 4 y 8, según el valor de los bits T1CKPS1 y T1CKPS0 del registro T1CON.

Tomando en cuenta estas características se pueden realizar retardos mayores sin utilizar los desbordes (hasta cierto límite), dependiendo de la base de tiempo del microcontrolador.

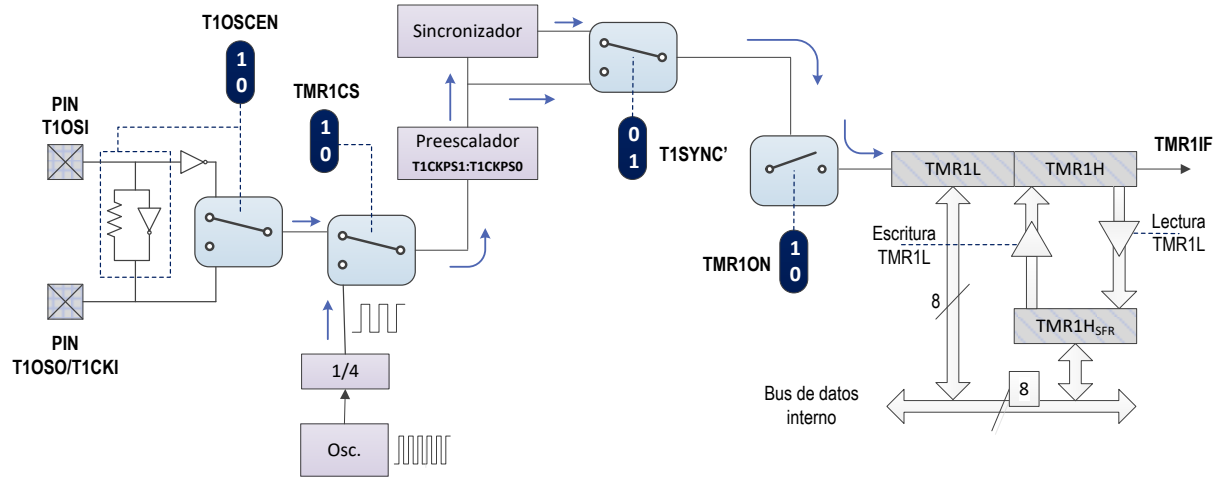


Figura 8.4: Arquitectura interna del módulo temporizador 1 (timer 1)

9. Edite en MPLAB un archivo ASM con el programa fuente en ensamblador que aparece en la figura 8.5.

10. Elabore el proyecto correspondiente y proceda a ensamblar el programa en la forma acostumbrada.

```
; ********************************************************
; Sistemas Digitales III p2016
; Programa que implementa un frecuencímetro en el PIC18F4550
; empleando el timer0 (modo temporizador) y el timer1 (modo
; contador). La frecuencia se despliega en los puertos D y B
; Fecha: 27/IV/2020
; Autor: ERA
; ********************************************************
      LIST  P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX HEX

;********************************************************
;               BITS DE CONFIGURACION
;********************************************************
;********       Configuracion del Oscilador     **********
;   Osc interno, RA6 como pin, USB usa Osc EC
      CONFIG  FOSC    = INTOSCIO_EC

;********      Bits de configuración mas usados     *********
      CONFIG  PWRT    = ON    ; PWRT habilitado
      CONFIG  BOR     = OFF   ; Brown out deshabilitado
      CONFIG  WDT     = OFF   ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON    ; MCLR como entrada de reset
      CONFIG  PBADEN  = OFF   ; PB como entradas digitales
      CONFIG  LVP     = OFF   ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF   ; los bloques del codigo de
      CONFIG  CP1     = OFF   ; programa no protegidos
      CONFIG  CPB     = OFF   ; Boot sector no protegido

; ********************************************************
; Variables para el programa
      cblock      20h
      CONT_DESB
      endc

;********************************************************
; Vector de Reset.
      ORG         0x0000
      goto        INICIO            ; Ve al inicio del programa pral

;********************************************************
; Vector de interrupcion de alta prioridad
      ORG         0x0008
      goto        ISR_INT_TIMER0    ; Ve al inicio del programa pral
```

Figura 8.5: Programa en lenguaje ASM para el experimento con el temporizador 1.

```
;*************************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
      movlw       B'01110010'
      movwf       OSCCON            ; Oscilador interno a 8 MHz
      return

;*************************************************************
; Subrutina que configura PB y PD como salida digital
CONF_PUERTOS
      clrf        LATB, 0           ; Limpia Lacths del puerto B
      clrf        LATC, 0           ; Limpia Lacths del puerto
      clrf        LATD, 0           ; Limpia Lacths del puerto D
      movlw       0Fh
      movwf       ADCON1, 0         ; Todos los pines como I/O dig
      clrf        TRISD, 0          ; Todo el PD como salidas
      clrf        TRISB, 0          ; Todo el PB como salidas
      bsf         TRISC, RC0, 0     ; RC0 como pin de entrada
      return

;*************************************************************
; Subrutina que configura el Timer0
CONF_TIMER0
      bcf         T0CON, TMR0ON     ; Timer0 apagado
      bcf         T0CON, T08BIT     ; Timer0 en modo 16 bits
      bcf         T0CON, T0CS       ; Timer0 en modo temporizador
      bcf         T0CON, PSA        ; Preescalador 1:2
      bcf         T0CON, T0PS0
      bcf         T0CON, T0PS1
      bcf         T0CON, T0PS2
      movlw       .20
      movwf       CONT_DESB, 0      ; CONT_DESB <- 20
      movlw       3Ch               ; Establece valor inicial
      movwf       TMR0H, 0          ; para un retardo
      movlw       0B0h              ; base = 50ms
      movwf       TMR0L, 0          ; de TMR0L y TMR0H
      bsf         T0CON, TMR0ON     ; Timer0 encendido
      return

;*************************************************************
; Subrutina que configura el Timer1 en modo contador
; sin preescalador y con la señal de entrada no sincronizada
CONF_TIMER1
      movlw       83h
      movwf       T1CON
      return

;*************************************************************
; Subrutina que configura la interrupcion del timer0 sin prioridad
CONF_INTERRUPS
      bcf         INTCON, TMR0IF, 0
      bsf         INTCON, TMR0IE, 0
```

Figura 8.5: Programa en lenguaje ASM para el experimento con el temporizador 1 (cont.)

```
      bsf         INTCON, GIE, 0
      return

;*****************************************************************
; Rutina de servicio de interrrupcion (ISR)
; Se activa aprox cada 50 ms debido al desborde del timer0
; Cada 20 desbordes actualiza en el PD y PB el valor de la frecuencia
; de la señal RC0/T1OSO/T1CKI
ISR_INT_TIMER0
      movlw       3Ch               ; Establece valor inicial
      movwf       TMR0H, 0          ; de TMR0L y TMR0H
      movlw       0B0h              ; para un retardo
      movwf       TMR0L, 0          ; base = 50ms
      decfsz      CONT_DESB, f, 0   ; Decrementa desbordes
      bra         AUN_NO            ; Desbordes = 0? No
      movlw       .20
      movwf       CONT_DESB, 0      ; CONT_DESB <- 20
      movff       TMR1L, LATD       ; Frecuencia (parte baja)
      movff       TMR1H, LATB       ; Frecuencia (parte alta)

      clrf        TMR1H, 0          ; Borrar los valores
      clrf        TMR1L, 0          ; previos de frecuencia
AUN_NO
      bcf         INTCON, TMR0IF, 0 ; Apaga bandera del TIMER0
      retfie



;*****************************************************************
; Programa principal
INICIO
      rcall CONF_BASE_TIEMPO
      rcall CONF_PUERTOS
      rcall CONF_TIMER0
      rcall CONF_TIMER1
      rcall CONF_INTERRUPS
      bra         $

      END
```

Figura 8.5: Programa en lenguaje ASM para el experimento con el temporizador 1 (cont.)

11. Implemente el circuito de la figura 8.6, descargue el programa al PIC y ejecútelo. Verifique el funcionamiento del circuito viendo el comportamiento de los leds. Conecte la terminal de salida TTL del generador de funciones al pin ***RC0*** a un canal del osciloscopio (debidamente calibrado) y encienda ambos. A continuación, elija la escala adecuada en el generador y varíe la frecuencia hasta que se observe en el osciloscopio que el periodo de la señal sea de 10 ms.

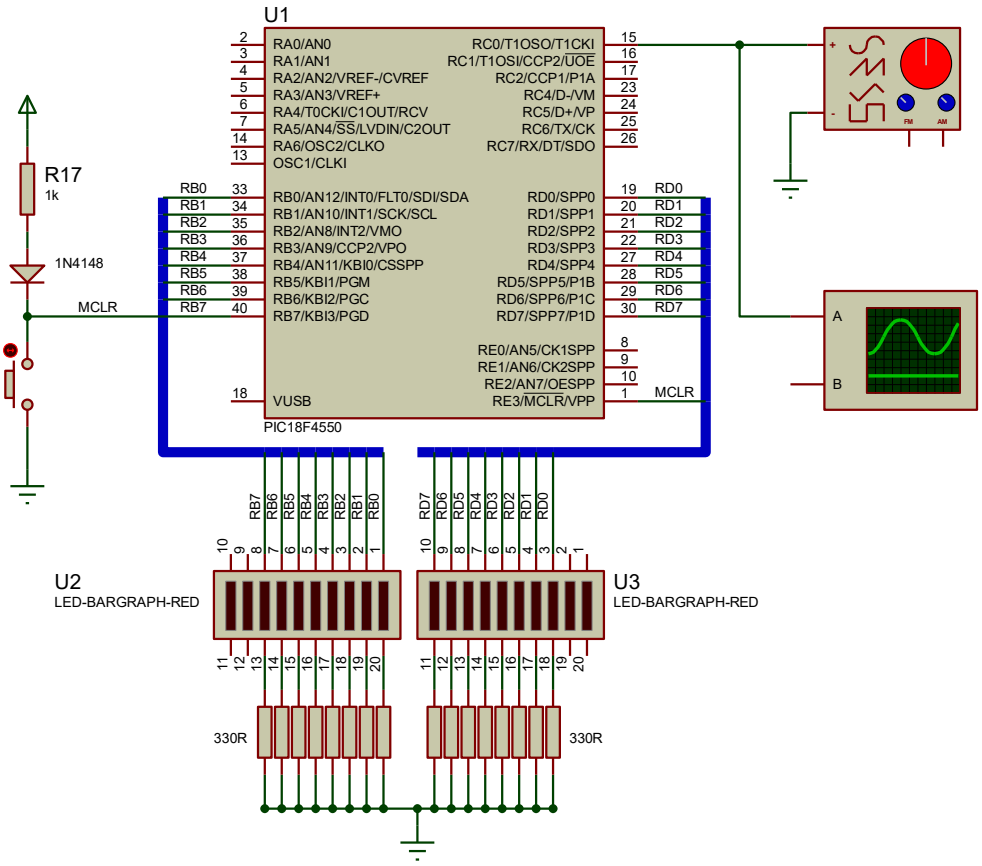


Figura 8.6: Circuito experimental para el temporizador 1.

12. Observe que el programa principal está formado por 6 líneas y básicamente lo que hace mandar las subrutinas que configuran: la base de tiempo a 1 microsegundo, los puertos D y B como salidas digitales, el Timer0 en modo temporizador de 16 bits, el Timer1 en modo contador asíncrono de pulsos externos y las interrupciones del Timer0. Hecho esto, el programa se queda en un ciclo ocioso en espera de la interrupción periódica de 50 ms de TMR0.

Durante el transcurso de las interrupciones de TMR0, el temporizador TMR1 recibe impulsos de reloj provenientes del generador de funciones, lo cual ocasiona que la cuenta de sus registros ($TMR1_H$ y $TMR1_L$) incrementen su valor. Se realiza un tiempo de muestreo con el TMR0 de 50 ms.

Cada vez que se llevan a cabo los 20 desbordes del temporizador TMR0 o su equivalente a un segundo, el PIC generará una interrupción y realizará las tareas programadas en la rutina de servicio. Esto es, precargará el valor de inicio de TMR0 en caso de no haber transcurrido el tiempo de un segundo y mostrar el valor de los registros $TMR1_H$:TMR_L, por los puertos B y D respectivamente y borrar la bandera del Timer1.

¿Qué valor muestran los leds conectados a los puertos B y D?. Considere que los puertos despliegan un “word” donde el bit RB0 es el MSB y RD0 es el LSB.

Valor de los puertos: _____

Convierta el valor hexadecimal anterior y compárelo con el valor mostrado en la carátula del generador de funciones. ¿Cómo son entre sí?

_____

_____

¿Existe alguna diferencia entre los valores mostrados en los leds y el mostrado por el generador de funciones?

_____

_____

Describa ¿a qué se debe esto?

Modifique el valor de la frecuencia de entrada al pin *RC0*, encuentre el rango de frecuencias en el que los valores mostrados en los leds son aproximados a los mostrados en la carátula del generador de funciones.

¿Cuál fue el rango obtenido? Escriba el valor máximo y mínimo.

_____

¿A qué se debe que el rango de medición sea limitado?

_____

¿En qué forma se podría incrementar el rango de medición? Justifique su respuesta.

_____

_____

Si incrementa demasiado el valor de frecuencia es posible que el resultado mostrado por el PIC sea mucho menor al valor real, ¿Cuál es la razón de esto?, ¿Cómo se puede evitar este efecto?

13. Del archivo fuente anterior, dentro de la subrutina que configura al Timer1, localice la línea de código:

```
movlw     87h
movwf     T1CON
```

sustitúyala ahora por la siguiente:

```
movlw      83h
movwf      T1CON
```

Almacene el archivo, reensamble el proyecto, descargue el programa al PIC y ejecútelo.

¿Cuál es el comportamiento de los leds ahora? ¿Varió la exactitud del resultado mostrado en los leds?

Exactamente ¿qué fue lo que hizo la nueva línea de código?

¿De qué manera influye en el funcionamiento del programa?

Varíe el rango de frecuencia de entrada en el pin RC0, ¿cómo es en comparación con el punto 12?

¿A qué se debe este comportamiento? Justifique su respuesta.

14. Analizando la rutina de servicio de interrupción del Timer0 (ISR_INT_TIMER0), puede notarse que línea de código

```
bcf        INTCON, TMR0IF, 0;
```

es esencial para el correcto funcionamiento del programa. Explique la razón.

15. Corrobore su respuesta eliminando del programa las líneas de código a la que hace referencia el punto anterior, reensamble el proyecto, reprograme el PIC y ejecútelo. ¿Qué es lo que sucede ahora?

_____

_____

ACTIVIDADES COMPLEMENTARIAS

Investigue y conteste los siguientes cuestionamientos.

1. ¿Cuáles son los tiempos de temporización máximos, sin considerar desbordes, que se pueden obtener de cada uno de los temporizadores?

2. Explique, ¿si es relevante o no la secuencia en que los registros de los temporizadores de 16 bits son escritos o leídos?

3. ¿Por qué los temporizadores de 16 bits tienen doble buffer en la parta alta de su registro? ¿cuál su función?

4. ¿Con cuál de los temporizadores disponible se puede implantar un reloj de tiempo real? Justifique su respuesta.

TEMA 9

EL MÓDULO CCP1 DEL PIC18F4550

INTRODUCCIÓN

Los microcontroladores PIC18F4550 disponen de dos módulos de Captura/Comparación/PWM (CCP1 y CCP2) que, en conjunto con los temporizadores, permite realizar de forma sencilla las tareas de medición de tiempo y frecuencia, y generación de señales digitales.

Los módulos CCP tienen 3 modos de funcionamiento, que se describen a continuación:

1. Modo captura (**capture**). Permite capturar el valor que tiene el registro de un temporizador (TMR1 o TMR3) cuando ocurre un evento especial en las terminales asociadas al CCP para captura (**RC2** para **CCP1** o **RC1** para **CCP2**).
2. Modo comparación (**compare**): Permite comparar el valor de 16 bits del TMR1 o TMR3 con un valor previamente definido ya sea en el par de registros **$CCPRL2_H$:$CCPR2_L$** o en el **$CCPRL1_H$:$CCPR1_L$**
3. Modo PWM: Permite generar señales digitales moduladas en ancho de pulso que salen por la terminal asociada al CCP (RC2 para CCP1 o RC1 para CCP2).

Mediante el experimento 9 podrá identificar cada uno de los periféricos que conforman al módulo CCP1 (Capture/Compare/PWM), configurar adecuadamente los registros de control del módulo CCP1 para que funcione en modo **Compare** y **PWM**.

La siguiente lista describe el material y equipo que se requiere:

Cantidad	Descripción
1	Microcontrolador PIC18F4550
3	*Push-bottons*
1	Resistencia de 1 kΩ
1	Resistencia de 10 kΩ
1	Diodo 1N4148
1	Tableta experimental

- Dispositivo programador compatible con PIC18F4550.
- Fuente de alimentación de CD.
- Osciloscopio.

- Computadora que tenga instalado el MPLAB® X versión v5.0 o superior y software para dispositivo programador de PIC's.

Como preparación para realizar el experimento 9 se recomienda leer previamente todo el documento.

EXPERIMENTO 9

DESARROLLO

I. Módulo CCP1 en modo Compare

1. Edite el programa mostrado en la figura 9.1.

```
; ********************************************************
; Sistemas Digitales III p2016
; Programa en ASM para generar una señal cuadrada
; empleando el módulo CCP1 en modo compare con interrupciones
      LIST        P=18F4550
      INCLUDE     <P18F4550.INC>
      RADIX       HEX
; ********************************************************
;                 BITS DE CONFIGURACION
; ********************************************************
;     CONFIGURACION DEL OSCILADOR
      CONFIG FOSC = INTOSCIO_EC     ;Osc interno, RA6 como pin E/S
;********      Bits de configuración mas usados     *********
      CONFIG  PWRT    = ON    ; PWRT habilitado
      CONFIG  BOR     = OFF   ; Brown out deshabilitado
      CONFIG  WDT     = OFF   ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON    ; MCLR como entrada de reset
      CONFIG  PBADEN  = OFF   ; PB como entradas digitales
      CONFIG  LVP     = OFF   ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF
;*********      Bits de proteccion      *****************
      CONFIG  CP0     = OFF   ; los bloques del codigo de
      CONFIG  CP1     = OFF   ; programa no protegidos
      CONFIG  CPB     = OFF   ; Boot sector no protegido

; ********************************************************
      CBLOCK      20h
      FLAG
      ENDC
; ********************************************************
; Vector de Reset.
      ORG         0x0000            ; Salta al inicio
      goto        INICIO            ; del programa pral
; ********************************************************
; Vector de interrupcion
      ORG         0x0008
      goto        ISR_CCP1          ; Ve a la ISR
```

Figura 9.1: Programa en ensamblador que genera una señal cuadrada usando el módulo CCP1 en modo comparación.

```
; ********************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
      movlw       B'01100010'       ; Oscilador interno
      movwf       OSCCON            ; a 4 MHz
      return

; ********************************************************
; Subrutina que configura el pin RC2 como salida digital
CONF_PUERTOS
      bcf         LATC, RC2, 0
      bcf         TRISC, RC2, 0
      return

; ********************************************************
; Subrutina para configurar el Timer1 en modo
; temporizador de 16 bits con preescalador de 1:8
CONF_TIMER1
      clrf        TMR1H, 0
      clrf        TMR1L, 0
      movlw       B'10110001'
      movwf       T1CON, 0
      return

; ********************************************************
; Subrutina que configura el modo compare del CCP1
; con interrupciones por comparacion exitosa,
CONF_CCP1
      bcf         RCON, IPEN, 0     ; Int. sin prioridad
      bsf         INTCON, PEIE, 0   ; Activa ints. periféricos
      bcf         PIR1, CCP1IF, 0   ; Limpia bandera del CCP1
      bsf         PIE1, CCP1IE, 0   ; Activa int por comparacion
      movlw       03h
      movwf       CCPR1H, 0
      movlw       0E8h
      movwf       CCPR1L, 0         ; CCP1R = _______ (dec)
      movlw       B'00001001'       ; RC2 = 1, cuando
      movwf       CCP1CON, 0        ; TMR1 = CCP1R
      return
; ********************************************************
; Rutina de servicio de interrupcion debido
; a comparación exitosa de TMR1 = CCPR1
ISR_CCP1
      clrf        TMR1H, 0
      clrf        TMR1L, 0
      btfss       FLAG, 0, 0
      bra         PRIMERO
      bra         SEGUNDO
PRIMERO
      movlw       b'00001000'       ; RC2 = 0, cuando
      movwf       CCP1CON, 0        ; TMR1 = CCP1R
      incf        FLAG, f, 0
      bra         SALIR
```

Figura 9.1: Programa en ensamblador que genera una señal cuadrada usando el módulo CCP1 en modo comparación (continuación).

```
SEGUNDO
      movlw       b'00001001'       ; RC2 = 1, cuando
      movwf       CCP1CON, 0        ; TMR1 = CCP1R
      clrf        FLAG, 0
SALIR
      bcf         PIR1, CCP1IF      ; Limpia la bandera
      retfie

; ********************************************************
;Rutina principal
INICIO
      rcall       CONF_BASE_TIEMPO
      rcall       CONF_PUERTOS
      rcall       CONF_TIMER1
      rcall       CONF_CCP1
      clrf        FLAG, 0
      bsf         INTCON, GIE, 0
      bra         $

      END
```

Figura 9.1: Programa que genera una señal cuadrada usando el módulo CCP1 en modo comparación (continuación).

2. Elabore el proyecto correspondiente y proceda a ensamblar el programa en la forma acostumbrada.

3. Implemente el circuito de la figura 9.2, descargue el programa al PIC y ejecútelo. Verifique el funcionamiento del circuito viendo la señal en el osciloscopio.

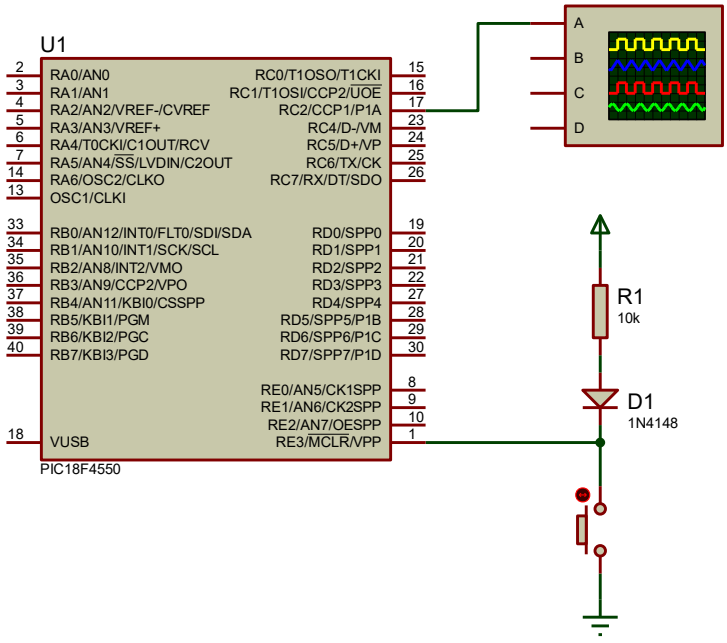


Figura 9.2: Circuito de experimentación para el CCP1 en modo capture

4. Mida el periodo de la señal ¿De cuánto es? _____

5. Modifique los valores que se cargan al CCPR1H y al CCPR1L de tal forma que la señal tenga un periodo de 1 ms (aproximado), sin modificar el valor del preescalador. Ponga sus cálculos que se realicen para lograr esta tarea.

_____

_____

6. Modifique los valores que se cargan al CCPR1H y al CCPR1L de tal forma que la señal tenga un periodo de 1 ms (exacto), modificando el preescalador.

_____

7. Del archivo fuente anterior, dentro de la rutina de servicio a interrupción (ISR), localice la línea de código

```
PIR1.CCP1IF=0;
```

Elimínela y almacene el archivo, recompile el proyecto, descargue el programa al PIC y ejecútelo. ¿Cuál es el comportamiento de la señal ahora?

_____

_____

8. Investigue el modo t*oggle* en la salida por comparación y utilice dicha ventaja para reducir código en la rutina de servicio a interrupción. ¿Qué modificación se tendría que hacer en la configuración del CCP1?

_____

_____

¿Cómo quedaría la rutina de servicio a la interrupción?

_____

_____

II. Módulo CCP1 en modo PWM

9. Edite en MPLAB el programa fuente que aparece en la figura 9.3.

10. Elabore el proyecto correspondiente y proceda a compilar el programa en la forma acostumbrada.

11. Implemente el circuito de la figura 9.4, descargue el programa al PIC y ejecútelo. Verifique el funcionamiento del circuito viendo la señal en el osciloscopio.

12. Mida el periodo y el ancho de pulso de la señal ¿De cuánto es?

_____

Escriba la forma de cómo se obtuvieron los valores anteriores.

_____

_____

```
; ********************************************************
; Laboratorio de Sistemas Digitales III p2016
; Programa en ASM para generar una señal con
; ciclo de trabajo variable empleando el
; módulo CCP1 en modo PWM

; Autor: ERA
; ********************************************************

      LIST         P=18F4550
      INCLUDE      <P18F4550.INC>
      RADIX       HEX

;********************************************************
;               BITS DE CONFIGURACION
;********************************************************
;********       Configuracion del Oscilador     **********
;   Osc interno, RA6 como pin, USB usa Osc EC
      CONFIG  FOSC    = INTOSCIO_EC

;********      Bits de configuración mas usados     *********
      CONFIG  PWRT    = ON    ; PWRT habilitado
      CONFIG  BOR     = OFF   ; Brown out deshabilitado
      CONFIG  WDT     = OFF   ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON    ; MCLR como entrada de reset
      CONFIG  PBADEN  = OFF   ; PB como entradas digitales
      CONFIG  LVP     = OFF   ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF
      CONFIG  XINST   = OFF

;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF   ; los bloques del codigo de
      CONFIG  CP1     = OFF   ; programa no protegidos
      CONFIG  CPB     = OFF   ; Boot sector no protegido
```

Figura 9.3: Programa en ensamblador que genera una señal PWM.

```
; Vector de Reset.
      ORG         0x0000            ; Salta al inicio
      goto        INICIO            ; del programa pral

;*********************************************************
; Vector de interrupcion
      ORG         0x0008
      goto        ISR_PB            ; Ve a la ISR

; *********************************************************
; Subrutina que configura la base de tiempo del MCU
CONF_BASE_TIEMPO
      movlw       B'01100010'       ; Oscilador interno
      movwf       OSCCON            ; a 4 MHz
      return

; *********************************************************
; Subrutina que configura el PB como entrada digital
; y el pin RC2 como salida digital
CONF_PUERTOS
      movlw       0x0F
      movwf       ADCON1, 0
      clrf        LATB, 0
      setf        TRISB, 0
      bcf         INTCON2, 7 ;
      bcf         LATC, RC2, 0
      bcf         TRISC, RC2, 0
      return

; *********************************************************
; Subrutina que configura la interrupcion por
; cambio en el nibble alto del PB
CONF_INT_PB
      bcf         RCON, IPEN, 0     ; Int. sin prioridad
      bsf         INTCON, RBIE, 0   ; Activa int. por PB
      movf        PORTB, w,0        ; Primer paso para
      bcf         INTCON, RBIF, 0   ; borrar bandera RBIF
      return

; *********************************************************
;Subrutina que configura el modo compare del CCP1
;con interrupciones por comparacion exitosa,
CONF_CCP1
      movlw       0FAh              ; Periodo = ______ms
      movwf       PR2, 0
      movlw       7Dh                     ;th = ____ms
      movwf       CCPR1L, 0
      movlw       b'00001100'       ; CCP1 en modo PWM
      movwf       CCP1CON, 0
      movlw       b'00000100'       ; TMR2 ON con
      movwf       T2CON             ; preescalador 1:1
```

Figura 9.3: Programa en ensamblador que genera una señal PWM (continuación).

```
; ********************************************************
;Rutina de servicio de interrupcion
;debido a cambio de nivel en nibble alto de PB
ISR_PB
      btfsc       PORTB, 6          ; RB6 = 0
      bra         SIGUE
      bra         INCREMENTA        ; Si
SIGUE
      btfsc       PORTB, 7          ; RB7 = 0
      bra         SALIR             ; No
      bra         DECREMENTA
INCREMENTA
      incf        CCPR1L, f, 0
      incf        CCPR1L, f, 0
      bra         SALIR
DECREMENTA
      decf        CCPR1L, f, 0
      decf        CCPR1L, f, 0
SALIR
      bcf         INTCON, RBIF
      retfie

;************************************************
;Rutina principal
INICIO
      rcall       CONF_BASE_TIEMPO
      rcall       CONF_PUERTOS
      rcall       CONF_INT_PB
      rcall       CONF_CCP1
      bsf         INTCON, GIE, 0
      bra         $

      END
```

Figura 9.3: Programa en ensamblador que genera una señal PWM (continuación).

13. Presione varias veces el *push-button* conectado en RB0. ¿Qué es lo que sucede con la señal?

_____

_____

14. Continúe pulsando y escriba lo que sucede cuando la anchura de pulso sobrepasa el periodo.

_____

_____

15. Repita los pasos anteriores, pero ahora pulsando el *push-button* conectado a RB1. Explique lo que sucede con la señal al observarla en el osciloscopio.

_____

_____

16. En términos de tiempo ¿De cuánto es el incremento o decremento del ancho de pulso de la señal al pulsar los *push-button*?

_____

_____

Escriba matemáticamente ¿cómo obtuvo los valores anteriores?

_____

_____

17. ¿Cuál es la resolución en bits para el ejemplo realizado en la figura 9.3?

_____

_____

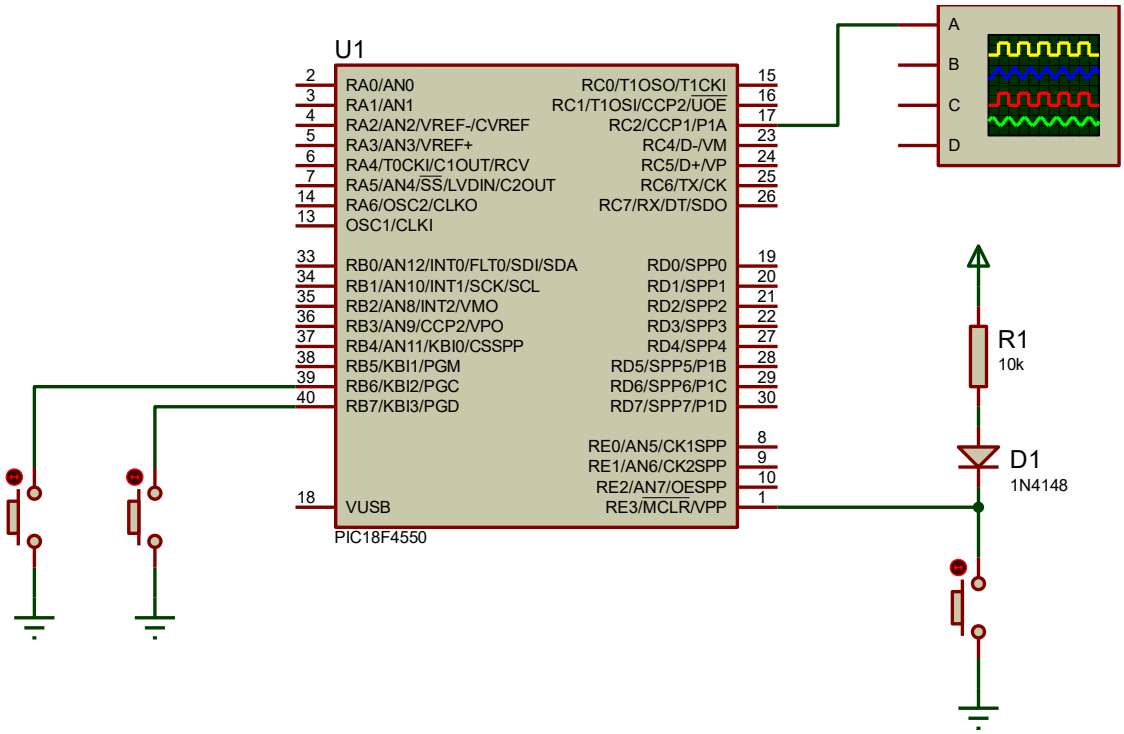


Figura 9.4: Circuito de experimentación para el CCP1 en modo PWM.

18. ¿A qué se debe que los cambios del ejercicio anterior sean como se observan? Explique en términos de la resolución del módulo PWM.

_____

_____

19. Analizando lo planteado en los incisos 17 y 18, explique con sus palabras, ¿qué es la resolución en el módulo CCP funcionando en modo PWM?

_____

_____

20. ¿De qué manera se afecta la resolución del modo PWM en función de la frecuencia de salida de la señal PWM y la base del tiempo del oscilador del MCU? Exponga la fórmula de resolución dada por el fabricante y justifique su respuesta.

_____

_____

_____

_____

ACTIVIDADES COMPLEMENTARIAS

Investigue y conteste los siguientes cuestionamientos.

1. ¿De qué forma se pueden asociar los módulos CCP's con temporizadores diferentes?

2. ¿Es posible asociar a pines diferentes la acción programada a una salida CCPx?

3. ¿Cuáles serían de forma general los pasos para emplear los CCP's en caso de medir tiempos de señales digitales?

4. ¿A qué se refiere el dicho de que los módulos CCP's pueden emplearse como "interrupciones por software"?

TEMA 10
CONVERTIDOR ANALÓGICO A DIGITAL (ADC) DEL PIC18F4550

INTRODUCCIÓN

En la actualidad los sistemas digitales que se encargan de controlar variables físicas como la temperatura, humedad, presión, flujo etc. Tienen incorporados convertidores de analógico a digital (ADC) y de digital a analógico (DAC). En la figura 10.1 se muestra un diagrama a bloques de las partes que conforman un sistema digital.

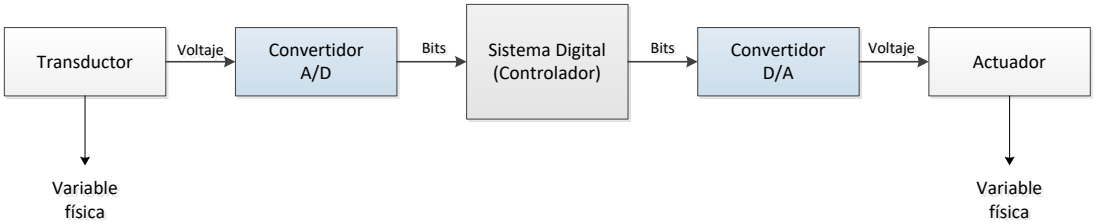


Figura 10.1: Sistema de procesamiento digital de variables físicas.

Cualquier información que se desea introducir a un sistema digital primero debe trasladarse a algún formato binario para que el microcontrolador pueda procesarlo. Esto puede ser ya sea en forma numérica (para ejecutar cálculos) o en su forma de cadena de caracteres para su almacenamiento y visualización. El bloque que realiza el proceso inverso (DAC) no es un periférico común que pueda estar disponible en los MCU's de propósito general. Lo anterior aunado a que cada vez son más los actuadores que tienen entradas de naturaleza digital que permiten manipularlo directamente de un MCU o microprocesador.

Para el caso del MCU PIC18F4550, tiene en su haber un módulo ADC de 10 bits que funciona con la técnica de aproximaciones sucesivas. Hay 13 entradas (canales) analógicos disponibles en forma múltiplexada por medio de los bits de control **CHS3:CHS2:CHS1:CHS0,** tal como puede visualizarse en la figura 10.2. Este periférico también puede simularse tanto en MPLAB® como en Proteus ®.

Asimismo, es posible configurar en dicho ADC los voltajes de referencia que determinan el valor de mínima escala (**VREF-**) y el de plena escala (**VREF+**). Con lo que, la conversión de un valor de voltaje analógico de entrada (**Vin**) en el rango de **VREF-** a **VREF+** producirá un valor equivalente binario (**D**) en el rango de **0 a 2^n-1**, donde **n** es la resolución en bits del convertidor (10, para el PIC18F4550).

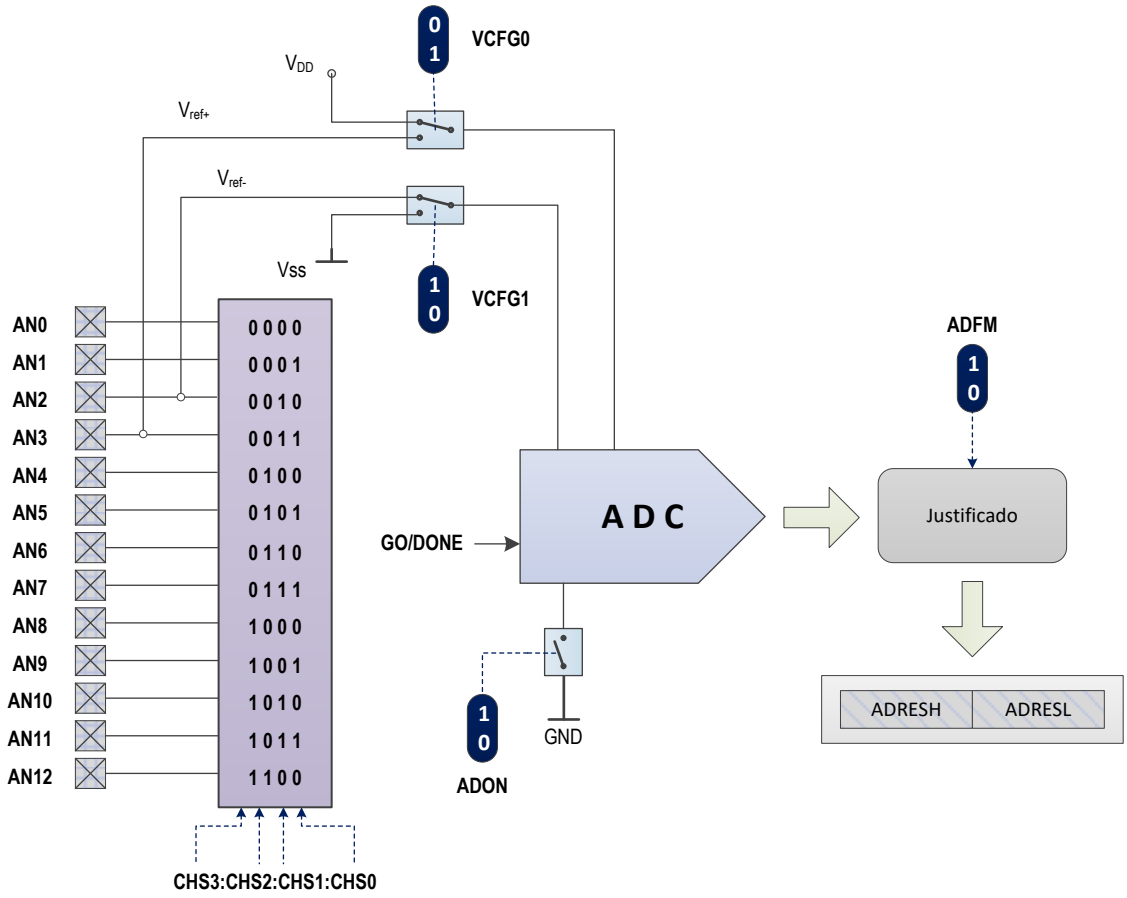


Figura 10.2: Arquitectura del módulo convertidor A/D del MCU PIC18F4550.

Como la relación entre escalas es lineal, una regla de tres proporciona la relación entre el voltaje analógico de entrada $(Vin)_{)}$ y el valor digital (D) obtenido por el ADC.

$$\frac{D}{2^n - 1} = \frac{V_{in} - V_{REF-}}{V_{REF+} - V_{REF-}} \qquad \text{ec. (10.1)}$$

Con la elección más común: $V_{REF}+ = V_{DD} = 5v$, $V_{REF}- = V_{ss} = 0V$, y como n = 10, se obtiene:

$$D = \frac{1023}{5} V_{in} = 204.6 V_{in} \qquad \text{ec. (10.2)}$$

De donde puede inferirse que cuando Vin varía en todo su rango, desde 0 hasta 5V, el valor obtenido D lo hace del mismo modo de 0 a 1023.

Cabe resaltar que existen dos parámetros relevantes que son necesarios para configurar correctamente el ADC interno del PIC18F4550: El tiempo de adquisición (T_{AD}) y la selección de la frecuencia del reloj de conversión. En forma general, cuando se usa una frecuencia de oscilación (F_{OSC}) de 8 MHz en el MCU, se debe configurar $4T_{AD}$ y un reloj de conversión 8 veces menor a F_{OSC}.

Con el experimento 10 se pretende entender el uso del ADC interno del PICF184550, sus registros de control y configuración. Se configurará el ADC interno en modo sondeo para digitalizar un canal analógico y para operar en modo interrupción para digitalizar un canal analógico y al mismo tiempo ejecutar otra tarea.

Los materiales y equipo se enlistan a continuación:

Cantidad	Descripción
1	Dispositivo programador de MCU's PIC
1	Microcontrolador PIC18F4550
1	Barra de leds.
1	Led
1	*Push-button*.
11	Resistencias de 330 Ω.
1	Resistencias de 4.7 kΩ.
1	Diodo 1N4148
1	Tableta experimental

- Multímetro.
- Osciloscopio.
- Fuente de alimentación de CD.
- Generador de funciones
- Computadora con MPLAB® X versión v5.0 o superior y software para dispositivo programador de PIC's instalado.

Se recomienda leer previamente todo el documento del experimento 10, llevar implementado en un *protoboard* el circuito de la figura 10.2, editar en ensamblador los códigos listados en la práctica e investigar en el manual de referencia los registros involucrados en la configuración del ADC del PIC18F4550.

EXPERIMENTO 10

DESARROLLO

I. El ADC en modo sondeo y canal simple

1. Edite el programa listado en la figura10.3, el cual configura el ADC para convertir el canal analógico 1 sin usar interrupciones (modo sondeo). El resultado de la conversión (10 bits) se envía en formato binario a los leds conectados en el puerto B (2 MSB) y D (8 LSB).

```
;***********************************************************
; Sistemas Digitales III p 2016
; Programa que configura el ADC interno del PIC18F4550
; en modo sondeo convirtiendo el canal analógico 1 (AN1)
; Autor: ERA
;***********************************************************
      LIST        P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX       HEX
;***********************************************************
;               BITS DE CONFIGURACION
;***********************************************************
      CONFIG  FOSC    = INTOSCIO_EC  ;Osc interno, RA6 como pin E/S

;********      Bits de configuración mas usados     **********
      CONFIG  PWRT    = ON           ; PWRT habilitado
      CONFIG  BOR     = OFF          ; Brown out deshabilitado
      CONFIG  WDT     = OFF          ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON           ; MCLR como entrada de reset
      CONFIG  PBADEN  = OFF          ; PB como entradas digitales
      CONFIG  LVP     = OFF          ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF          ; Opción de debug apagado
      CONFIG  XINST   = OFF          ; Instrucciones ext apagado

;*********      Bits de proteccion      *****************
      CONFIG  CP0     = OFF          ; Los bloques del codigo de
      CONFIG  CP1     = OFF          ; programa no estan protegidos
      CONFIG  CPB     = OFF          ; Sector Boot no esta protegido
;***************************************************************
; Vector de Reset.
      ORG         0x0000
      goto        INICIO      ; Ve al inicio del programa principal

;***************************************************************
; Vector de interrupcion
      ORG         0x0008
      goto        INICIO      ; Ve al inicio del programa principal
```

Figura 10.3: Programa en ensamblador para el funcionamiento del ADC en modo sondeo.

```
;*****************************************************************
; Subrutina que configura la base de tiempo
CONF_CLK
      movlw       0x72
      movwf       OSCCON      ; Oscilador interno a 8 MHz
      return

;*****************************************************************
; Subrutina que configura los puertos B, D y el RA2 como salidas
; digitales y el pin RA1 como entrada analógica
CONF_PUERTOS
      clrf        LATA
      clrf        LATB
      clrf        LATD
      movlw       0x0D
      movwf       ADCON1      ; Configura RA0 y RA1 como pines analógicos
      setf        TRISA       ; Configura como entrada al puerto A
      clrf        TRISB       ; Configura como salida al puerto B
      clrf        TRISD       ; Configura como salida al puerto D
      return

;*****************************************************************
; Subrutina que configura el ADC en modo sondeo (sin interrupciones
; y convirtiendo el canal analogico 1 (AN1)
CONF_ADC
      movlw       0x04        ; Elije canal AN1
      movwf       ADCON0
      movlw       0x91        ; Configura justificado a la derecha,
      movwf       ADCON2      ; 4 Tad y Frc/8 (parametros para Fosc 8 MHz)
      bsf         ADCON0,0    ; Enciende el ADC
      return

;*****************************************************************
;       PROGRAMA PRINCIPAL
INICIO
      call        CONF_CLK
      call        CONF_PUERTOS
      call        CONF_ADC
OTRA
      bsf         ADCON0,1    ; Inicia una conversión A/D
ESPERA
      Btfss       PIR1,ADIF         ; Espera a que termine conversion
      bra         ESPERA
      movff       ADRESL,LATD       ; Despliega 8 bits LSB en puerto D
      movff       ADRESH,LATB       ; Despliega 2 bits MSB en puerto B
      bcf         PIR1,ADIF         ; Borra bandera ADIF
      bra         OTRA              ; Repite proceso de conversion

      END
```

Figura 10.3: Programa en ensamblador para el funcionamiento del ADC en modo sondeo (continuación).

2. Ensamble el programa en la forma acostumbrada y verifique que no existan errores inspeccionando el archivo de error. Posteriormente proceda a descargar el archivo HEX al al PIC.

3. Implemente el circuito mostrado en la figura 10.4 y energícelo.

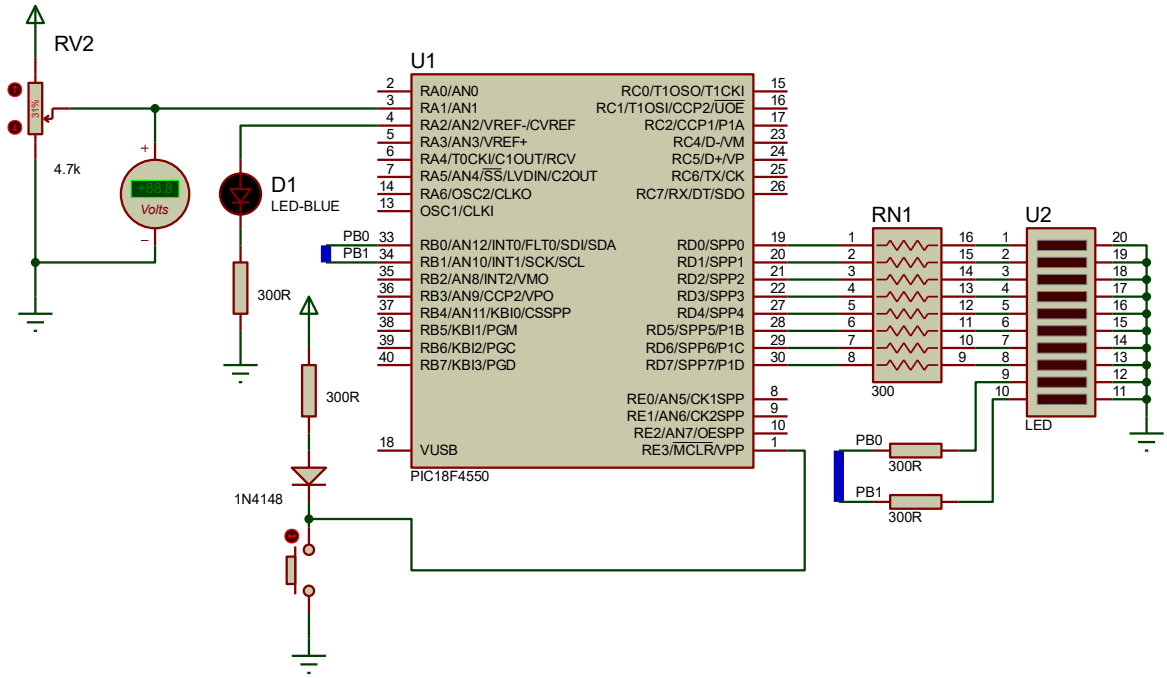


Figura 10.4: Circuito experimental para el ADC del PIC18F4550.

4. Con ayuda del multímetro y una fuente de alimentación variable, ajuste hasta obtener un V_{in}=1 V. Energice el PIC y ejecute el programa ¿Cuál es lo que se observa en la pantalla de la PC?

_____

5. Repita el experimento anterior, variando el V_{in} con los valores que se indican en la tabla 10.1 y anotando el valor correspondiente de su conversión. Para calcular el resultado teórico de conversión usar la ecuación (10.2)

Tabla 10.1: Relación entre Vin y la salida digital del ADC.

Vin	Resultado de la conversión (teórico)	Resultado de la conversión (Binario)	Resultado de la conversión (Hex)
0 V.			
0.5 V			
1 V			
1.5 V			
2 V			
2.5 V			
3 V			
3.5 V			
4 V			
4.5 V			
5 V			

7. Con los valores obtenidos en la tabla 10.1, grafique la función de transferencia del ADC.

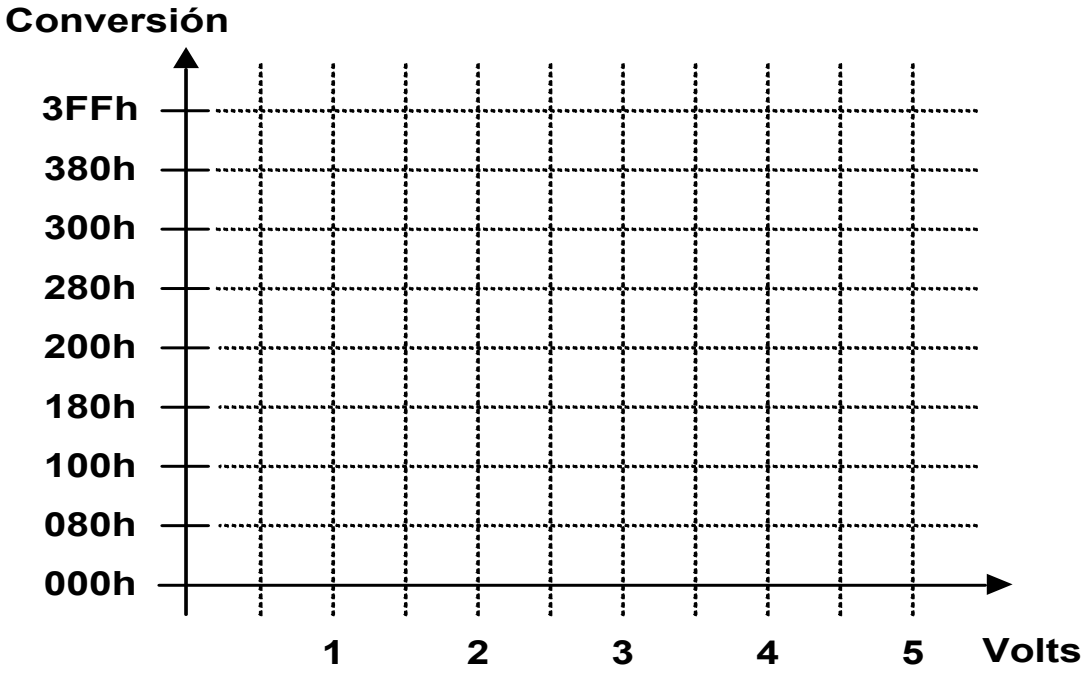


Figura 10.5: Gráfica de la función de transferencia del ADC.

II. El ADC en modo interrupción y canal simple

Cuando se requiere que el proceso de conversión sea más eficiente y no genere tiempos muertos que esclavicen al MCU, se debe configurar al ADC en modo interrupción. Esto evitará que el MCU tenga que sondear el bit de bandera **ADIF** y por lo tanto, puede programarse para realizar otras tareas mientras el ADC realiza la conversión, con la plena certeza de que cuando esta finalice se generará una interrupción.

8. Edite el programa listado en la figura10.6, el cual configura el ADC para convertir el canal analógico usando interrupciones generadas por fin de conversión (bandera **ADIF**). Al mismo tiempo el MCU hará parpadear un led conectado en el pin **RA2**. El resultado de la conversión (10 bits) se envía en formato binario a los leds conectados en el puerto B (2 MSB) y D (8 LSB). Analícelo y compárelo con el anterior. ¿Cuáles son las diferencias en la estructura del programa?

_____

_____

¿Qué diferencias existen en la rutina que configura al ADC?

_____

```
;*********************************************************
; Programa que configura el ADC interno del PIC18F4550
; en modo sondeo convirtiendo el canal analógico 1 (AN1)
; Autor: ERA
      LIST        P = 18F4550
      INCLUDE     <P18F4550.INC>
      RADIX       HEX

;*********************************************************
;               BITS DE CONFIGURACION
      CONFIG  FOSC    = INTOSCIO_EC  ;Osc interno, RA6 como pin e/s

;********      Bits de configuración mas usados     **********
      CONFIG  PWRT    = ON          ; PWRT habilitado
      CONFIG  BOR     = OFF         ; Brown out deshabilitado
      CONFIG  WDT     = OFF         ; Watchdog deshabilitado
      CONFIG  MCLRE   = ON          ; MCLR como entrada de reset
      CONFIG  PBADEN  = OFF         ; PB como entradas digitales
      CONFIG  LVP     = OFF         ; Prog bajo voltaje apagado
      CONFIG  DEBUG   = OFF         ; Opción de debug apagado
      CONFIG  XINST   = OFF         ; Instrucciones ext apagado
;*********      Bits de proteccion      ******************
      CONFIG  CP0     = OFF         ; Los bloques del codigo de
      CONFIG  CP1     = OFF         ; programa no estan protegidos
      CONFIG  CPB     = OFF         ; Sector Boot no esta protegido
```

Figura 10.6: Programa en el funcionamiento del ADC en modo interrupción.

```
;*****************************************************************
; Vector de Reset.
      ORG         0x0000
      goto        INICIO            ; Ve al inicio del programa principal

;*****************************************************************
; Vector de interrupcion
      ORG         0x0008
      goto        ISR_ADC           ; Ve a la ISR del ADC

;*****************************************************************
; Configura el reloj interno a 2 MHz
CONF_CLK
      movlw       0x72
      movwf       OSCCON
      return
;*****************************************************************
; Subrutina que configura todo el puerto B y D como salidas
; digitales, el pin RA2 como salida digital y el pin RA1 como
; entrada analógica
CONF_PUERTOS
      clrf        LATA
      clrf        LATB
      clrf        LATD
      movlw       0x0D
      movwf       ADCON1      ; Configura RA0 y RA1 como pines analógicos
      bsf         TRISA, RA1  ; Configura como entrada a RA0
      bcf         TRISA, RA2  ; Configura como entrada a RA1
      clrf        TRISB       ; Configura como salida al puerto B
      clrf        TRISD       ; Configura como salida al puerto D
      bsf         LATA, RA2   ; Manda un 1 a RA2
      return

;*****************************************************************
; Configura el ADC interno en modo interrupción
; convirtiendo el canal analógico 1 (AN1)
CONF_ADC
      movlw       0x04        ; Elije canal AN1
      movwf       ADCON0
      movlw       0x91        ; Configura justificado a la derecha
      movwf       ADCON2      ; 4 Tad y Frc/8 (parámetros para Fosc 8 MHz)
      bsf         ADCON0, 0   ; Enciende el ADC
      bcf         PIR1, ADIF  ; Apaga bandera ADIF
      bsf         PIE1, ADIE  ; Activa mascara local del ADC
      bsf         INTCON, PEIE; Activa la máscara de periféricos
      return
```

Figura 10.6: Programa en el funcionamiento del ADC en modo interrupción (continuación).

```
;*************************   Retardo 1ms      *************************
; Subrutina (bloqueante) que ejecuta un retardo de 1 ms por software
RETARDO_UN_MS
      movlw       .249
OTRO  addlw       0xFF        ; Decrementa W
      btfss       STATUS,Z,.0 ¿ Es igual a 0?
      bra         OTRO        ; No => seguimos esperando
      return

;*****************************************************************
; Rutina de servicio de interrupcion debido a la bandera ADIF
ISR_ADC
      movff       ADRESL, LATD      ; Despliega 8 bits LSB en puerto D
      movff       ADRESH, LATB      ; Despliega 2 bits MSB en puerto B
      bcf         PIR1,ADIF         ; Borra bandera ADIF
      bsf         ADCON0, 1         ; Inicia una conversion A/D
      retfie

;*****************************************************************
;       PROGRAMA PRINCIPAL
INICIO
      call        CONF_CLK
      call        CONF_PUERTOS
      call        CONF_ADC

      bsf         ADCON0, 1         ; Inicia una conversión A/D
      bsf         INTCON, GIE       ; Activa mascara global de int.

BLINK
      btg         LATA, RA2         ; Invierte el nivel lógico de RA2
      rcall       RETARDO_UN_MS     ; retardo de 1 ms
      bra         BLINK

      END
```

Figura 10.6: Programa en el funcionamiento del ADC en modo interrupción (continuación).

9. Proceda a ensamblar y descargar el programa en el PIC en la forma acostumbrada.

10. Ejecute el programa en el circuito de la figura 10.2. ¿Existen diferencias existen en el funcionamiento con respecto al primer experimento del ADC?

_____

_____

Explique, ¿de qué forma el PIC logra ejecutar tanto la tarea de conversión como la de hacer parpadear el LED?.

_____

_____

ACTIVIDADES COMPLEMENTARIAS

1. ¿De qué forma se puede incrementar la resolución del ADC del PIC18F4550?

2. ¿Es posible usar una resolución 8 bits en el ADC?, explique.

3. ¿Teóricamente, ¿Cuál es la frecuencia máxima que puede tener la señal a convertir?

4. ¿De qué forma sería más eficiente hacer una rotación automática de los canales a convertir?

BIBLIOGRAFÍA

ÁNGULO J. M., ROMERO S., ÁNGULO I., Microcontroladores PIC Diseño Práctico de Aplicaciones segunda parte: PIC16F87X, PIC18FXXXX, 2ª edición, McGraw-Hill, España, 2006.

CUENCA, E. Martín, et al, Microcontroladores PIC, La Solución en un Chip, Paraninfo, España, 2001.

GALEANO G., Programación de Sistemas Embebidos en C, Alfaomega, 2011.

GARCíA E., Compilador C CCS y Simulador PROTEUS para Microcontroladores PIC. Alfaomega, 2008.

MICROCHIP TECNOLOGY INC., PIC16F882/883/884/886/887 *Datasheet*, Código del documento:DS41291F, 2009

MICROCHIP TECNOLOGY INC., PIC18F2455/2550/4455/4550 *Da*tasheet, Código del documento: DS39632E, 2009

MILAN VERLE, PIC Microcontrollers - Programming in C, Mikroelektronika; 1st edition (2009).

PALACIOS, E., et al, Microcontrolador PIC16F84, Desarrollo de proyectos, 2ª Ed., Alfaomega-Ra-Ma, España, 2006.

WILMSHURST T., Designing Embedded Systems With PIC Microcontrollers, Principles and applications, Newnes, 2007.

SOFTWARE DE APOYO

MPLAB X 5.0: Herramienta de programación y diseño de proyectos basados en microcontroladores PIC de Microchip.

BBK-precission: Software de interfaz para dispositivo programador universal BBK-precission-747-USB.

Notepad++ 7.8.7: Software libre para edición de texto y código fuente con soporte para varios lenguajes de programación.

PIC Kit 2.0: Software para descarga de programas de microcontroladores PIC.

Proteus professional v8.8: Interfaz de desarrollo y de simulación de circuitos electrónicos.

LISTA DE ACRÓNIMOS

ADC: Analog Digital Converter; Convertidor de Analógico a Digital.

ARES: Advanced Routing and Editing Software; Software de Edición y Enrutamiento Avanzado.

BSR: Bank Select Register; Registro Selector de Banco.

CAD: Computer Assisted Design; Diseño Asistido por Computadora.

CCP: Compare/Capture/PWM; Comparación/Captura/PWM.

CD: Current Direct; Corriente Directa.

DB: Define Byte; Define un Byte.

DT: Define Table; Define una Tabla.

E/S: Entrada/Salida.

F_{OSC}: Frecuencia de Oscilación

FSR: File Select Register; Registro Selector de Almacenamiento.

GIE: General Interrupt Enable; Habilitador General de Interrupción.

GND: Tierra Física.

I/O: Input/Output; Entrada/Salida.

INDF: Indirect File Register; Registro de Almacenamiento Indirecto.

ISIS: Intelligent Schematic Input System; Sistema Inteligente de Captura Esquemática.

ISR: Interrupt Routine Service; Rutina de Servicio de Interrupción.

GPR: General Purpose Register: Registro de Propósito General.

LSB: Less Significant Byte: Byte Menos Significativo.

LVP: Low Programming Voltaje; Voltaje bajo de programación.

MCLRE: Master Clear Enable; Habilitador de Reinicio Maestro.

MCU: Microcontrolador.

MSB: Most Significant Byte: Byte Más Significativo.

PCB: Printed Circuit Board; Circuito Impreso.

SFR: Specific File Register; Registro de Almacenamiento Específico.

VSM: Virtual System Modelling; Sistema de Modelado Virtual.

T_{AD}: Tiempo de Adquisición.

Experimentos prácticos en lenguaje ensamblador con el PIC18F4550

Se terminó de editar en Ciudad Obregón, Sonora; el 30 de junio de 2025, por la Oficina de Publicaciones del Instituto Tecnológico de Sonora. Fue puesto en línea para su disposición en el sitio https://www.itson.mx/publicaciones/ en la sección Ingeniería y tecnología.

RESUMEN

El libro **Experimentos Prácticos en Lenguaje Ensamblador con el PIC18F4550** es una guía técnica enfocada en la enseñanza y aplicación de la programación en ensamblador para microcontroladores PIC, especialmente el modelo PIC18F4550. A través de una serie de experimentos, proporciona un enfoque práctico para comprender el funcionamiento de los registros, periféricos y técnicas de optimización del código en sistemas embebidos.

El contenido inicia con una introducción a las herramientas de desarrollo, mostrando el uso de MPLAB X para la programación y simulación de proyectos, y posteriormente la integración con Proteus para pruebas en un entorno virtual. Luego, se aborda la configuración de los bits esenciales del PIC18F4550, explicando aspectos fundamentales como la selección del oscilador y la protección de memoria. Se presentan técnicas de direccionamiento indirecto para la manipulación de datos y el uso de tablas en memoria de programa, para la lectura y escritura de información. También se explora la decodificación de teclados matriciales mediante algoritmos como el *walking ones*, permitiendo una mejor gestión de entradas en sistemas digitales.

El libro cubre el manejo de interrupciones externas, enseñando la configuración de los pines del puerto B usados para la captura de eventos sin interferir con el flujo de ejecución principal. Se dedica un capítulo al funcionamiento de los temporizadores programables, detallando el uso del Timer0 y Timer1 en modo temporizador y contador para el control preciso de eventos de tiempo. Además, se examina el módulo CCP1 en sus modos de captura, comparación y PWM, ilustrando su aplicación en la generación de señales digitales moduladas.

Finalmente, se aborda el convertidor analógico-digital (ADC) del PIC18F4550, explicando su configuración, tanto en modo sondeo como el de interrupción, para la digitalización de señales analógicas, permitiendo su procesamiento en sistemas embebidos. Cada experimento proporciona una base sólida para el aprendizaje y desarrollo de proyectos, facilitando la comprensión de la arquitectura del PIC18F4550 y su aplicación en ingeniería electrónica y mecatrónica.