UNAM

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

Departamento:

Ingeniería

Asignatura:

Microcontroladores

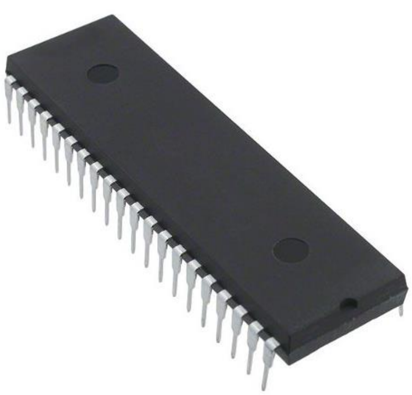
Sección:

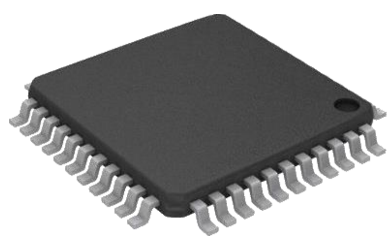
Electrónica

Clave de carrera: 130

Clave de asignatura: 1919

Manual de Laboratorio de Microcontroladores





Autores: M. en C. Leopoldo Martín del Campo Ramírez
Ing. Marcelo Bastida Tapia

Fecha de elaboración: enero 2015
Fecha de revisión: julio 2024
Semestre 2025-1

Laboratorio de Microcontroladores

Índice

Objetivos generales de la asignatura III

Objetivos del curso experimental III

Introducción III

Reglamento interno de laboratorios de electrónica IV

Criterios de evaluación del curso VI

Bibliografía VII

Introducción Guía de inicio de MPLAB X IDE
-Lenguaje ensamblador 1
-Simulación 7

Práctica 1 Manejo de señales multiplexadas
4.1. Multiplexación de señales.
-Microcontrolador PIC12F615 14

Práctica 2 Puertos del microcontrolador
4.2. Descripción funcional de terminales.
-Microcontrolador PIC16F887 19

Práctica 3 Temporizadores
5.3. Programación de microcontroladores.
-Microcontrolador PIC12F615 24

Práctica 4 Interrupciones
5.4. Programación de los registros internos.
-Microcontrolador PIC12F615 29

Práctica 5 Temporizadores y contadores
6.3. Elaboración de proyectos.
-Microcontrolador PIC16F887 34

Práctica 6 Control de motores a pasos
7.2. Diseño de interfaces de control de potencia eléctrica.
-Microcontrolador PIC16F887 40

Práctica 7 **Comparadores de voltaje**
7.5. Control con optoacopladores.
-Microcontrolador PIC16F628A 46

Práctica 8 **Convertidor Analógico – Digital**
8.1. Conversión Analógico Digital con microcontroladores.
-Microcontrolador PIC16F887 51

Práctica 9 **Modulación por ancho de pulso**
9.1. Modulación por ancho de pulso (PWM) empleando microcontroladores.
-Microcontrolador PIC16F628A 56

Práctica 10 **Comunicación serial – Protocolo I2C**
9.6. Otros recursos.
-Microcontrolador PIC16F887 61

Laboratorio de Microcontroladores

Prólogo

Objetivo general de la asignatura

- Al finalizar el curso el alumno conocerá y comprenderá la estructura y funcionamiento de los microcontroladores y podrá aplicar dichos elementos en la solución de problemas de control dedicado, así como la aplicación de diversas plataformas de desarrollo de sistemas electrónicos, de telecomunicaciones y mecatrónica, entre otros.

Objetivos del curso experimental

- Conocer la arquitectura interna básica de los microcontroladores
- Conocer las características de los periféricos que integran al microcontrolador.
- Realizar programas en lenguajes de bajo y alto nivel para solucionar problemas de diseño.
- Programar los microcontroladores y comprobar que las soluciones propuestas a los retos de diseño sean funcionales.
- Implementar sistemas basados en microcontroladores.

Introducción

Los circuitos digitales permiten la implementación de operaciones lógicas, combinacionales y secuenciales, mediante la interconexión de circuitos integrados (IC's) que contienen compuertas lógicas, flip-flop's y otros elementos. Sin embargo, conforme crecen las necesidades de diseño y la complejidad de cada aplicación, esta se vuelve una solución impráctica y costosa.

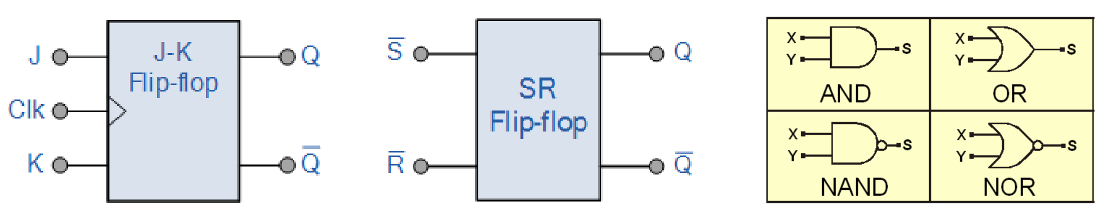


Figura I. Ejemplos de flip-flop's y circuitos combinacionales.

En la actualidad existe una serie de dispositivos conocidos como microcontroladores; estos reúnen dentro de un solo chip muchas de las características propias de una computadora, como pueden ser las memorias RAM y ROM o los puertos, además de otros elementos. El que se encuentren todos los elementos del microcontrolador dentro de un solo chip indica que son dispositivos de muy alta densidad de integración. Los microcontroladores suelen ser utilizados en aplicaciones de control debido a que utilizan de manera más eficiente la memoria de que disponen además de poseer un conjunto de instrucciones mucho más reducido en comparación con una computadora.

Otra característica que ha contribuido para que los microcontroladores hayan tomado la preferencia de muchos usuarios es su bajo costo pues, aunque varía dependiendo de la capacidad de cada modelo de microcontrolador, generalmente resultan ser muchas veces más baratos que otros medios de control como pueden ser las computadoras o los controladores lógicos programables (PLC's).

Los microcontroladores poseen una gran capacidad de adaptación a las distintas aplicaciones de control que se presentan en la industria, a pesar de que sus capacidades son limitadas. Esto lo compensan con la gran cantidad de modelos existentes, de los cuales se puede seleccionar aquel que cubra las necesidades del sistema a controlar.

En estas prácticas se busca desarrollar sistemas basados en microcontroladores explotando su capacidad de adquisición de datos, como pueden ser los puertos digitales y los convertidores analógicos – digitales, además de sus capacidades de control, como por ejemplo el control de motores de corriente directa y de motores de pasos, necesarios para la implementación de sistemas mecánicos industriales o de robots.

Finalmente, el alumno deberá comprender la necesidad de interacción entre las diferentes áreas de la Ingeniería para llevar a la práctica un sistema de control basado en microcontrolador. Entre estas áreas se distinguen:

- Electrónica Analógica
- Ingeniería de Control
- Control Digital
- Sistemas Digitales
- Computación
- Diseño de Software
- Electrónica de Potencia
- Mecánica
- Motores
- y algunas áreas adicionales.

Reglamento interno de laboratorios de electrónica

1. Queda estrictamente prohibido, al interior de los laboratorios

    a) Correr, jugar, gritar o hacer cualquier otra clase de desorden.
    b) Dejar basura en las mesas de trabajo y/o pisos.
    c) Fumar, consumir alimentos y/o bebidas.
    d) Realizar o responder llamadas telefónicas y/o el envío de cualquier tipo de mensajería.
    e) La presencia de personas ajenas en los horarios de laboratorio.
    f) Dejar los bancos en desorden y/o sobre las mesas.
    g) Mover equipos o quitar accesorios de una mesa de trabajo.
    h) Usar o manipular el equipo sin la autorización del profesor.
    i) Rayar y/o sentarse en las mesas del laboratorio.
    j) Energizar algún circuito sin antes verificar que las conexiones sean las correctas (polaridad de las fuentes de voltaje, multímetros, etc.).
    k) Hacer cambios en las conexiones o desconectar el equipo estando energizado.
    l) Hacer trabajos pesados (taladrar, martillar, etc.) en las mesas de trabajo.
    m) Instalar software y/o guardar información en los equipos de cómputo de los laboratorios.
    n) El uso de cualquier aparato o dispositivo electrónico ajeno al propósito para la realización de la práctica.
    o) Impartir clases teóricas, su uso es exclusivo para las sesiones de laboratorio.

2. Es responsabilidad del profesor y de los estudiantes revisar las condiciones del equipo e instalaciones del laboratorio al inicio de cada práctica (encendido, dañado, sin funcionar, maltratado, etc.). El profesor deberá generar el reporte de fallas de equipo o de cualquier anomalía y entregarlo al responsable de laboratorio o al jefe de sección.

3. Los profesores deberán de cumplir con las actividades y tiempos indicados en el "cronograma de actividades de laboratorio".

4. Es requisito indispensable para la realización de las prácticas que el estudiante:

   a) Descargue el manual completo y actualizado al semestre en curso, el cual podrá obtener en (http://olimpia.cuautitlan2.unam.mx/pagina_ingenieria/)
   b) Presente su circuito armado en la tableta de conexiones para poder realizar la práctica (cuando aplique), de no ser así, tendrá una evaluación de cero en la sesión correspondiente.
   c) Realizar las actividades previas y entregarlas antes del inicio de la sesión de práctica, de no ser así, tendrá una evaluación de cero en la sesión correspondiente.

5. Estudiante que no asista a la sesión de práctica de laboratorio será evaluado con cero.

6. La evaluación de cada sesión debe realizarse con base en los criterios de evaluación incluidos en los manuales de prácticas de laboratorio y no podrán ser modificados. En caso contrario, el estudiante deberá reportarlo al jefe de sección.

7. La evaluación final del estudiante en los laboratorios será con base en lo siguiente:

   a) (Aprobado) Cuando el promedio total de todas las prácticas de laboratorio sea mayor o igual a 6 siempre y cuando tengan el 90% de asistencia y el 80% de prácticas acreditadas con base en los criterios de evaluación.
   b) (No Aprobado) No cumplió con los requisitos mínimos establecidos en el punto anterior.
   c) (No Presentó) Cuando no asistió a ninguna sesión de laboratorio o que no haya entregado actividades previas o reporte alguno.

8. Profesores que requieran hacer uso de las instalaciones de laboratorio para realizar trabajos o proyectos, es requisito indispensable que las soliciten por escrito al jefe de sección. Siempre y cuando no interfiera con los horarios de los laboratorios.

9. Estudiantes que requieran realizar trabajos o proyectos en las instalaciones de los laboratorios, es requisito indispensable que esté presente el profesor responsable del trabajo o proyecto. En caso contrario no podrán hacer uso de las instalaciones.

10. Correo electrónico del buzón para quejas y sugerencias para cualquier asunto relacionado con los laboratorios (seccion_electronica@cuautitlan.unam.mx).

11. El incumplimiento a estas disposiciones faculta al profesor para que instruya la salida del infractor y en caso de resistencia, la suspensión de la práctica.

12. A los usuarios que, por su negligencia o descuido inexcusable, cause daños al laboratorio, materiales o equipo deberá cubrir los gastos que se generen con motivo de la reparación o reposición, indicándose en el reporte de fallas correspondiente.

13. Los usuarios de laboratorio que sean sorprendidos haciendo uso indebido de equipos, materiales, instalaciones y demás implementos, serán sancionados conforme a la legislación universitaria que le corresponda, según la gravedad de la falta cometida.

14. Los casos no previstos en el presente reglamento serán resueltos por el Jefe de Sección, de acuerdo con los lineamientos generales para el uso de los laboratorios en la Universidad Nacional Autónoma de México.

Instrucciones para la elaboración del reporte

Será necesario incluir en cada actividad previa y reporte de práctica una portada (obligatoria) que contenga la información mostrada en el ejemplo de la figura II.

Para la presentación del reporte se deberá cumplir con los requisitos indicados en cada una de las prácticas, incluyendo:

- Introducción
- Procedimiento experimental
- Tablas de datos
- Mediciones
- Gráficas
- Comentarios
- Observaciones
- Esquemas
- Diagramas
- Cuestionario
- Conclusiones
- Bibliografía

y en general todos los elementos solicitados dentro del desarrollo de la práctica.

U. N. A. M.
F. E. S. C.

Laboratorio de:	____________________
Grupo:	____________________
Profesor:	____________________
Alumno:	____________________
Nombre de Práctica:	____________________
No. de Práctica:	____________________
Fecha de realización:	____________________
Fecha de entrega:	____________________
Semestre:	____________________

Figura II. Información necesaria para la portada de actividades previas y reportes de prácticas.

Criterios de evaluación del curso

C1 (Criterio de evaluación 1): Actividades previas (investigaciones y simulaciones) (30%)
C2 (Criterio de evaluación 2): Escritura y compilación de códigos con comentarios (20%)
C3 (Criterio de evaluación 3): Habilidad para el manejo del equipo e interpretación correcta de las lecturas (10%)
C4 (Criterio de evaluación 4): Reporte entregado con todos los puntos indicados (40%)

Bibliografía

1) **Programación de sistemas embebidos en C**
Galeano, Gustavo
Alfaomega, 1º Edición
México, 2009
2) **Compilador C CCS y Simulador Proteus para microcontroladores PIC**
García, Eduardo
Alfaomega, 1º Edición
México, 2008
3) **Microcontroladores, Fundamentos y Aplicaciones con PIC**
Valdes Pérez, Fernando
Alfa Omega
México, 2007
4) **AVR microcontroladores. Configuración total de periféricos**
López Chau, Ausdrubal
Universidad Autónoma del Estado de México UAEM
México, 2006
5) **Atmega8 Data Sheet**
Atmel Corporation, 2013
6) **Embedded C Programming with the AVR Microcontroller**
R. H. Barnett, S. Cox, y L. O'Cull
Cengage Learning 2ND Edition
Clifton Park, NY, 2006
7) **Some Assembly Required: Assembly Language Programming with the AVRMicrocontroller**
T. S. Margush
CRC Press, 1ST Edition
Boca Raton, 2011

Laboratorio de Microcontroladores

Guía de inicio de MPLAB X IDE

El software libre MPLAB X IDE se define como un editor modular que permite seleccionar los distintos microcontroladores soportados, crear códigos de programación y compilarlos, además de permitir la grabación de dichos circuitos integrados en conjunto con un programador compatible.

El editor de MPLAB X IDE cuenta con una gran cantidad de opciones diferentes en cuanto a herramientas de compilación, tipo de archivo, tipo de procesador, etc., por lo que es complicado hacer uso de él sin alguna guía de inicio y explicación que sirva como apoyo. A continuación, se presenta una forma básica de crear un proyecto de programación haciendo uso de esta plataforma empleando lenguaje ensamblador. Esta guía se limita solamente a la manera de crear un proyecto y las configuraciones necesarias para comenzar a trabajar usando como ejemplo un programa básico.

NOTA: Es importante descargar e instalar el software gratuito de MPLAB X IDE desde el sitio oficial de Microchip y al final de la instalación se debe indicar que instale de manera adicional la herramienta XC8 para tener también la posibilidad de utilizar el lenguaje C, que será empleada más adelante.

Lenguaje ensamblador

En la figura 0.1 se presenta la pantalla de inicio de la plataforma de MPLAB X IDE donde aparecen señaladas las opciones para comenzar un nuevo proyecto.

Figura 0.1. Ventana de inicio de MPLAB X IDE v6.20.

Para comenzar se hace clic en el botón de ***nuevo proyecto,*** a continuación, se mostrará una ventana como la que se presenta en la figura A.2, se selecciona la categoría “Microchip Embedded” y el tipo de proyecto como “Application Project” y se hace clic en *Next*.

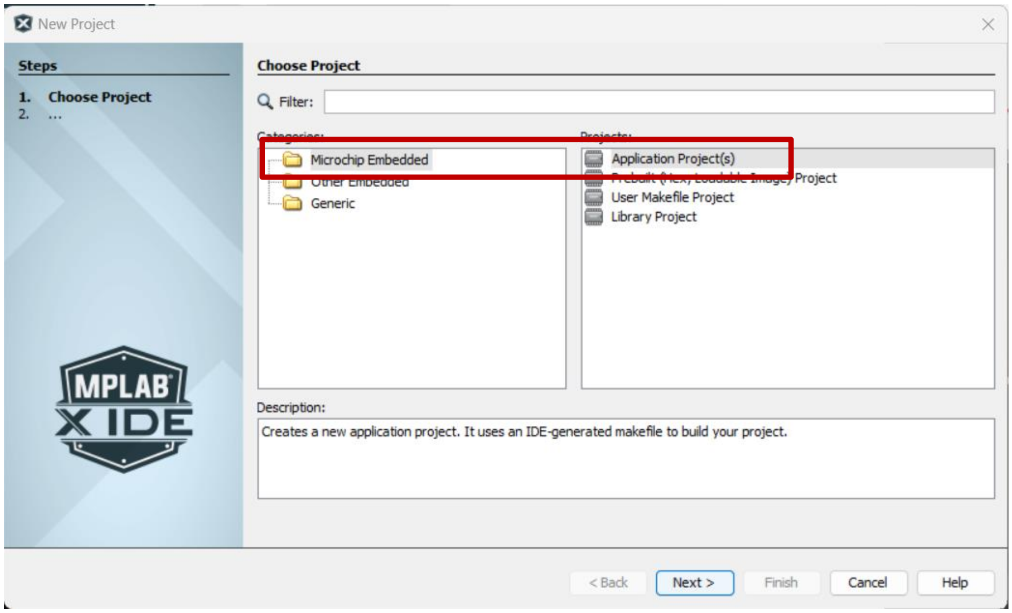


Figura 0.2. Ventana de selección del tipo de proyecto.

Una vez seleccionado el tipo de proyecto se indica el modelo del dispositivo para el cual esta destinado el programa que se va a realizar. Los dispositivos disponibles van desde microcontroladores PIC de gama base, hasta microcontroladores dsPIC, además de los microcontroladores AVR. En la figura 0.3 se muestra un ejemplo de la selección del dispositivo con el que se va a trabajar, en este caso el PIC12F628A. Se puede hacer una búsqueda por familia de dispositivos o buscar de manera directa el modelo deseado.

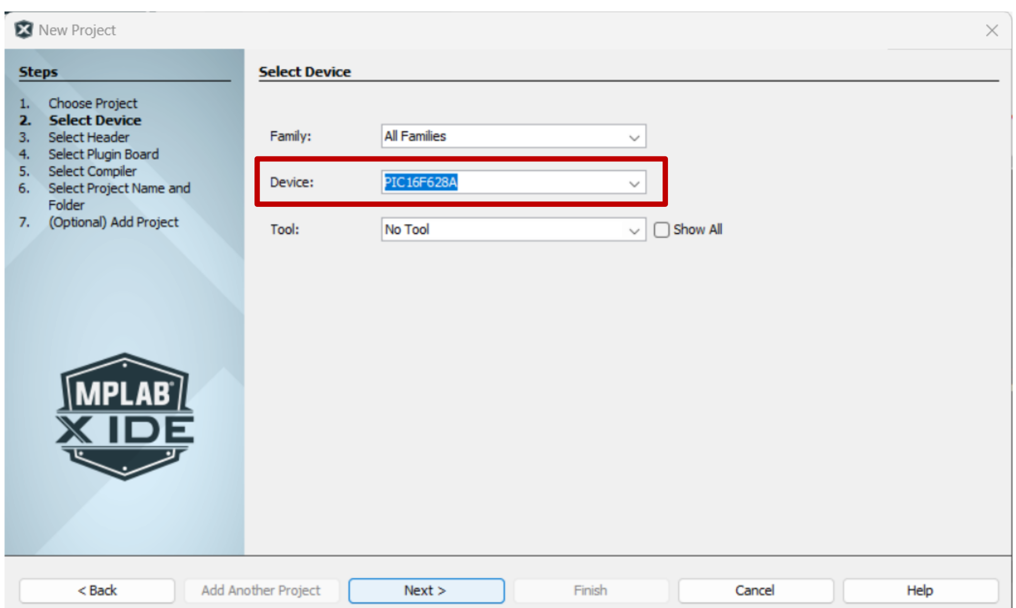


Figura 0.3. Selección del dispositivo.

En la ventana siguiente, selección del *Header* (encabezado), figura 0.4, no es necesario realizar ninguna acción y se puede dar clic en *Next* de manera inmediata.

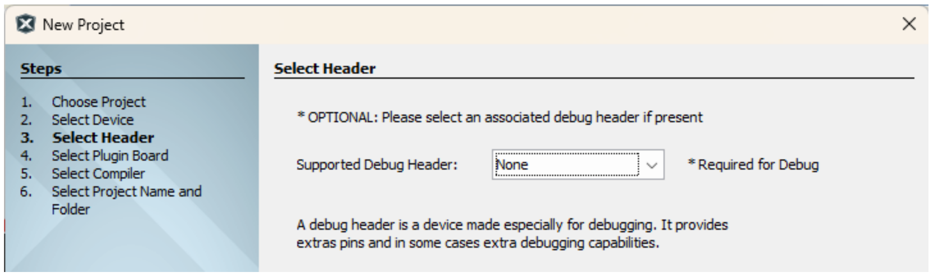


Figura 0.4. Selección del *Header*.

El siguiente paso es muy importante, por lo que se debe realizar correctamente. La ventana que aparece después de la de selección de *Header* es la ventana de **Selección del Compilador**. Aquí se debe elegir la herramienta para compilar el código del programa que se va a editar. En este caso, se va a realizar un programa en **lenguaje ensamblador (asm)** por lo que la herramienta seleccionada deberá ser aquella que sea compatible con dicho lenguaje.

Dependiendo de cada computadora, de la fecha de instalación y de las actualizaciones que se realicen, pueden estar instaladas varias herramientas de compilación, como se ve en la figura 0.5, solo es necesario elegir la apropiada. En la figura 0.5 se observa que se tienen 4 herramientas instaladas, se selecciona la herramienta de compilación **pic-as (v2.46)**, que en este caso es la herramienta para lenguaje ensamblador más reciente con la que se disponía.

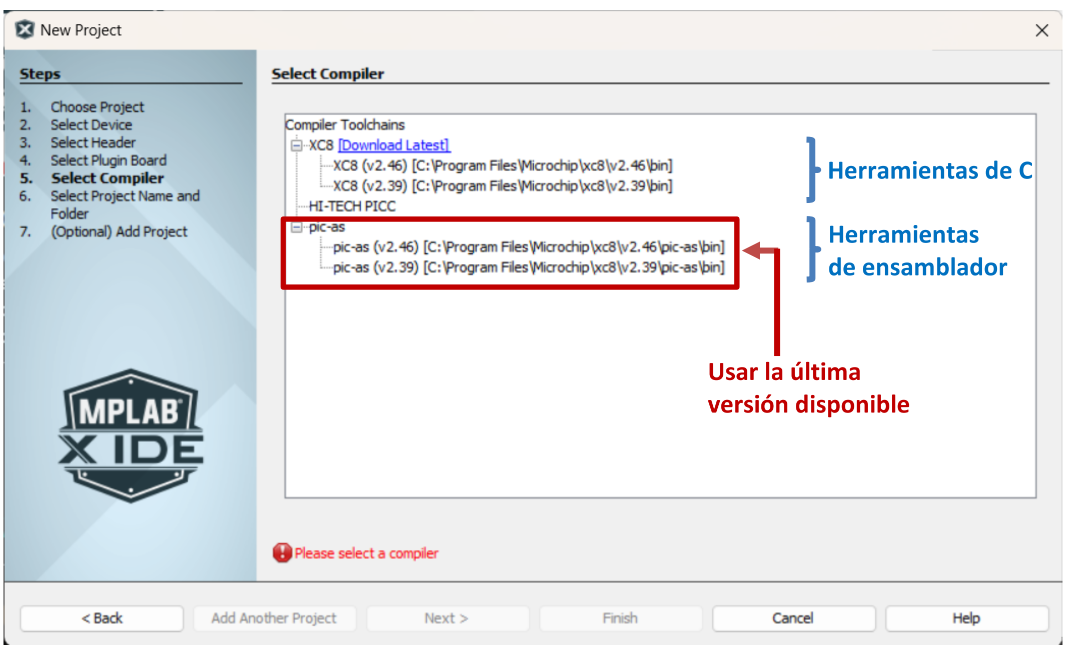
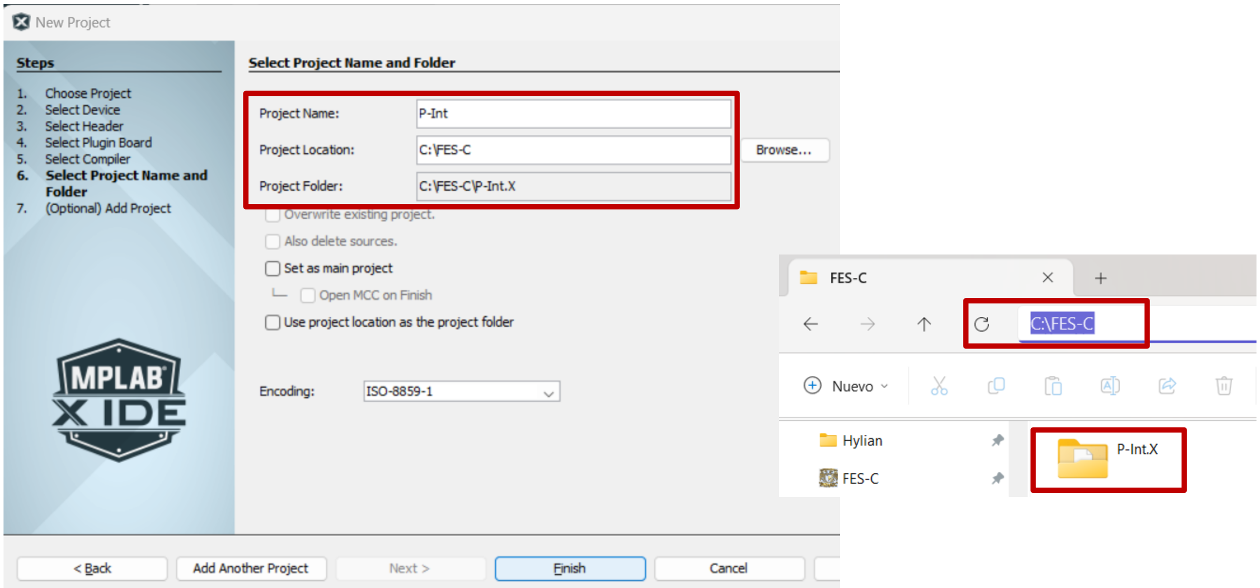


Figura 0.5. Selección del Compilador.

Para terminar la creación el proyecto se debe le debe de dar un nombre y asignar una dirección de memoria, en la figura 0.6(a) se muestra que el nombre dado al proyecto es **P-Int**, la ubicación de memoria elegida es en el disco **C:** en la carpeta llamada **FES-C** y, de manera automática, se creará una carpeta llamada **P-Int.X** donde se almacenarán todos los archivos generados del proyecto y su posterior compilación, como se muestra en la figura 0.6(b).

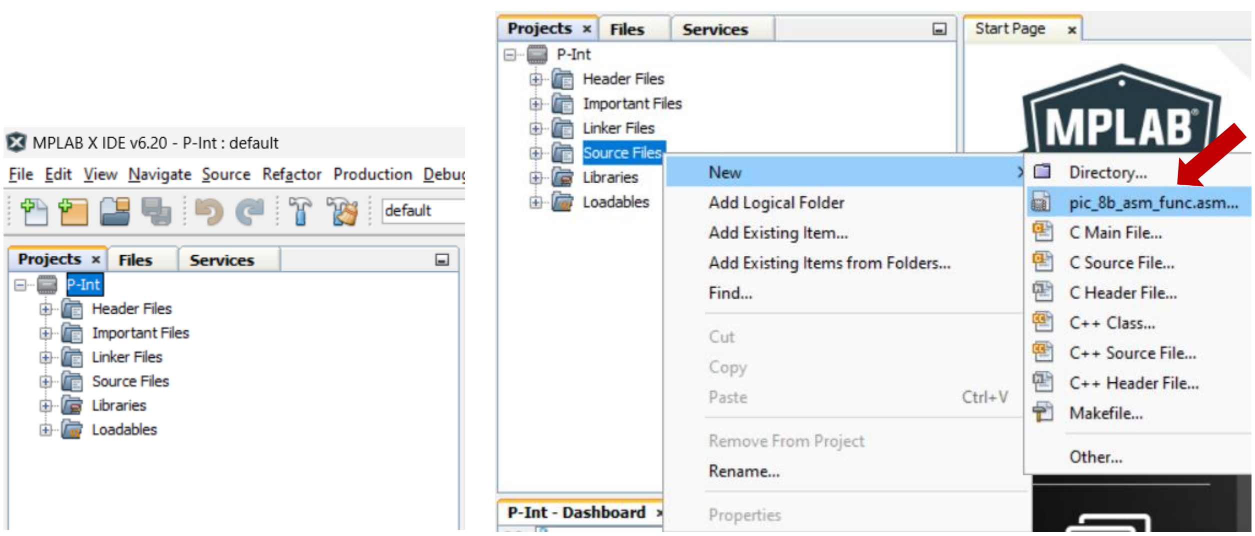


(a) (b)

Figura 0.6. Nombre y ubicación del proyecto.

Con el proyecto ya creado, solo resta agregar el archivo del código fuente (source code) donde se redactará y, posteriormente, se compilará el código.

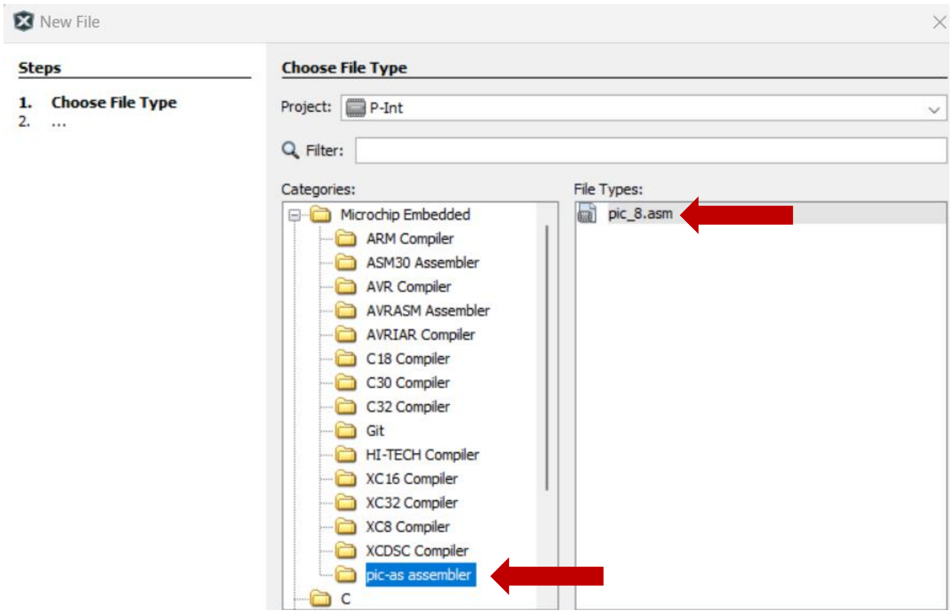
En la figura 0.7(a) se observa que el área de la ventana de proyecto ya cuenta con una serie de carpetas. Se debe de hacer clic derecho en la carpeta correspondiente al **Source Files** e indicar que se desea un nuevo archivo de tipo **pic_8b_asm_func.asm**, como se ve en la figura 0.7 (b).



(a) (b)

Figura 0.7. Archivo fuente para el proyecto.

NOTA: En caso de que no aparezca la opción ***pic_8b_asm_func.asm*** entonces se elige la opción *Other…* que abrirá otra ventana donde se puede elegir el tipo de archivo como aparece en la figura 0.7(c).



(c)

Figura 0.7. Forma alternativa de elegir el archivo fuente para el proyecto.

Al seleccionar el tipo de archivo fuente se abrirá una ventana donde se puede especificar el nombre del archivo y el tipo de extensión, dicho archivo se creará dentro de la carpeta del proyecto, como se observa en la figura 0.8(a) en los recuadros marcados.

El nombre puede ser asignado de manera automática por MPLAB X IDE o se puede asignar manualmente, **se recomienda asignar el mismo nombre que se le dio al proyecto para evitar confusiones**. La extensión del archivo a seleccionar es ***"s"***, como se ve en la figura 0.8. No es necesario modificar ninguna otra opción y simplemente se hace clic en *Finish*.

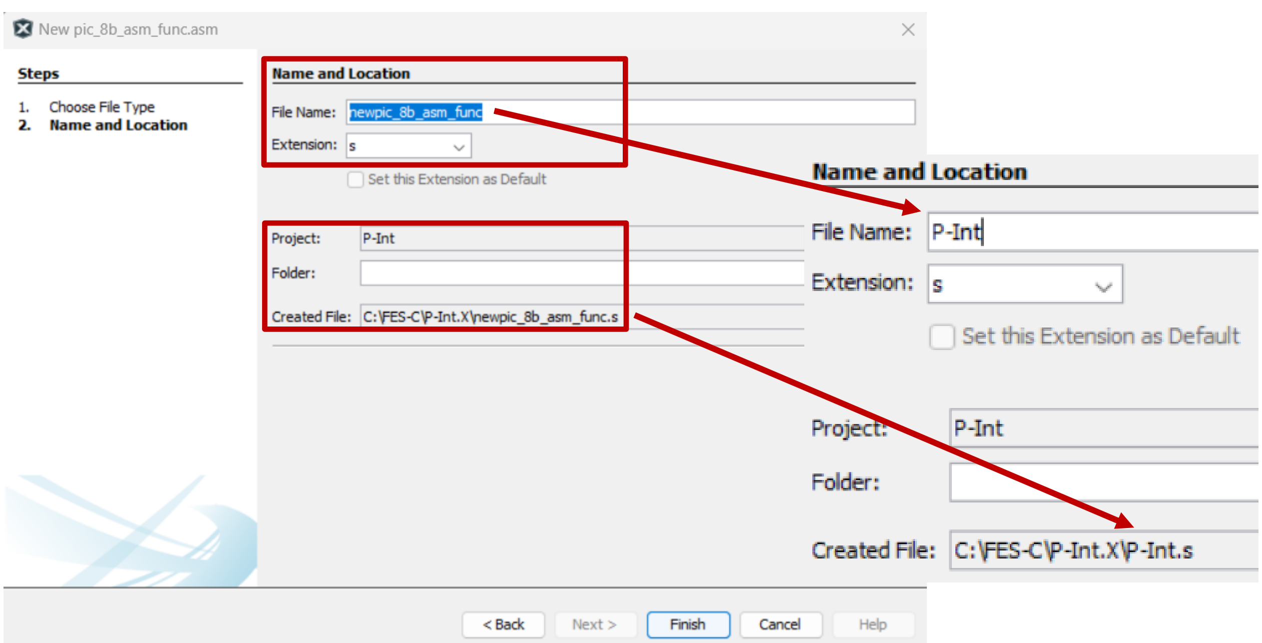


Figura 0.8. Nombre y extensión del archivo fuente para el proyecto.

A partir de este punto, ya es posible comenzar a redactar el código del programa que se desee. La ventana del código que aparecerá por defecto se muestra en la figura 0.9. Se observa que aparecen escritas múltiples líneas de texto y código en la ventana, la primera línea es la directiva *#include<xc.inc>*, seguida de una gran cantidad de *comentarios*, que se distinguen porque cada línea comienza con ‘ ; ’ y, por último, un conjunto de comandos que van de la línea 15 a la 21 del código.

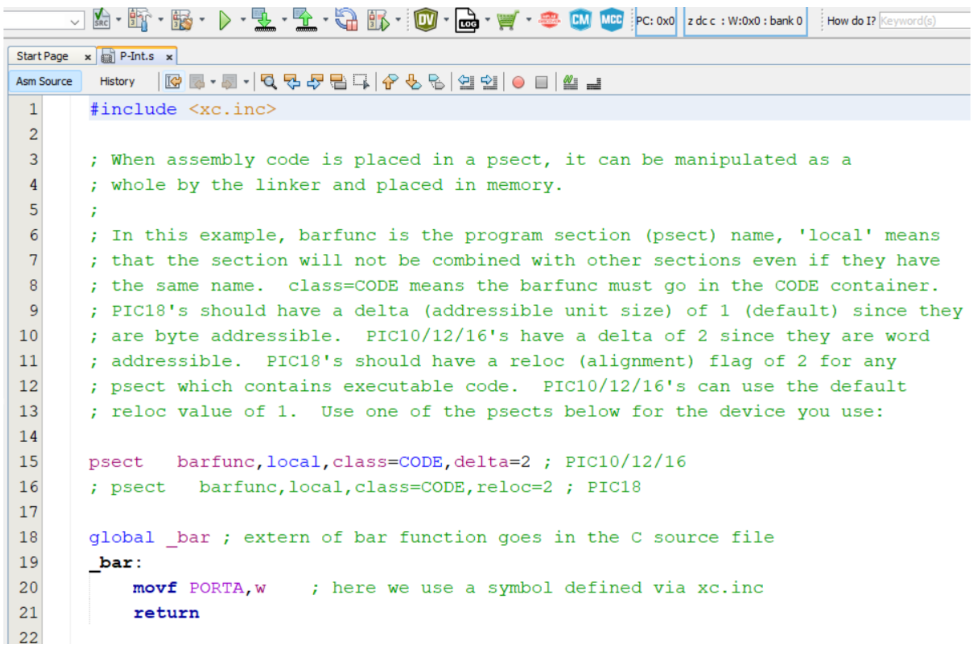


Figura 0.9. Ventana del código fuente del proyecto.

Para evitar confusiones y hacer lo más simple el código se empezará ***BORRANDO TODO*** y a continuación se comenzará desde cero. En la figura 0.10 se muestra que lo primero que hay que hacer es escribir la directiva ***PROCESSOR***, que sirve para indicar al compilador el modelo de microcontrolador con el que se esta trabajando, en este ejemplo el PIC16F628A, el cual ayuda a definir todas las opciones internas del dispositivo para que no sea necesario definirlas por parte del usuario.

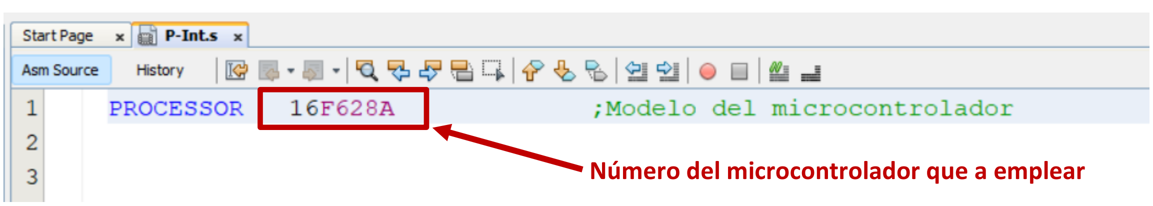


Figura 0.10. Definición del dispositivo en la ventana del código fuente.

El siguiente paso es usar la herramienta para la creación de la palabra de configuración. Para acceder a esta opción solo es necesario dirigirse al menú *Production* y entrar en la opción *Set Configuration Bits*, como se ve en la figura 0.11.

Lo anterior abrirá una pequeña ventana en la parte inferior de la pantalla donde se pueden seleccionar las opciones deseadas de la palabra de configuración de acuerdo con las capacidades del dispositivo indicado, por ejemplo, la selección de la fuente de reloj, las opciones de protección y reinicio del dispositivo y las opciones de seguridad, entre otras. Esto se muestra en la figura 0.12.

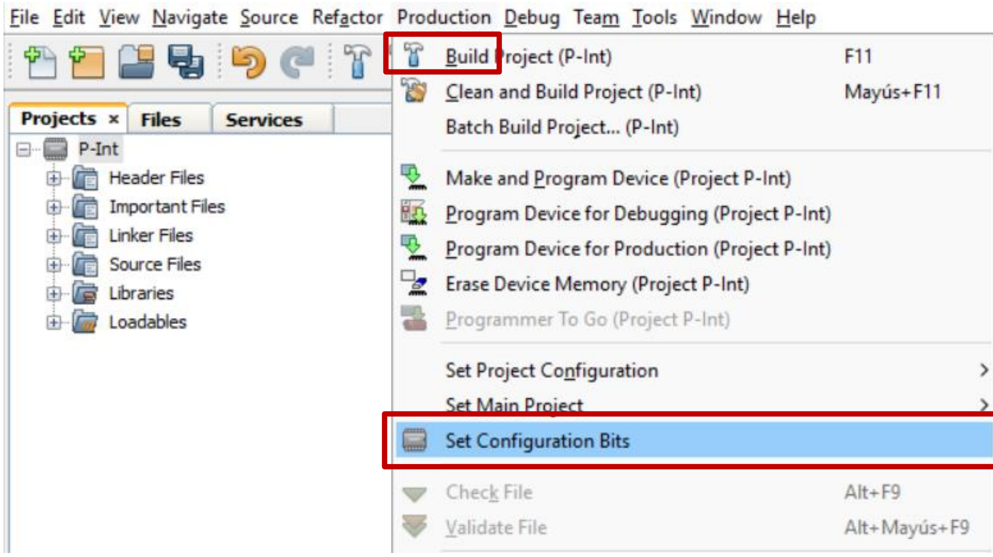


Figura 0.11. Herramienta para generar la palabra de configuración.

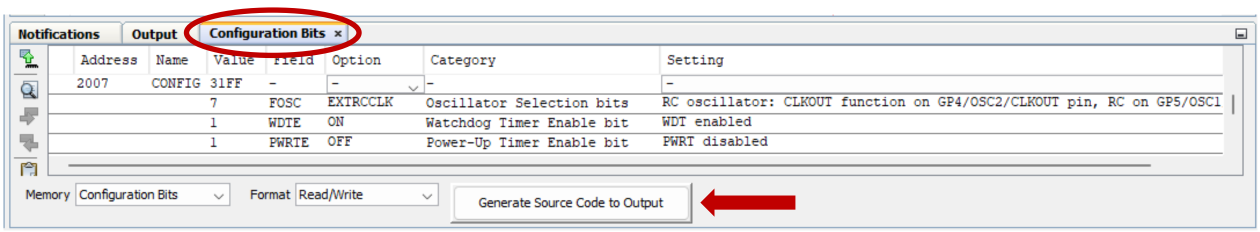


Figura 0.12. Generación del código correspondiente a la palabra de configuración.

Para este ejemplo, se van a seleccionar las opciones de: fuente de reloj interna, WDT apagado, PWRT habilitado, MCLR presente en la terminal RA5, BOD habilitado, programación de bajo voltaje deshabilitada y protección de la memoria de datos y el código apagados.

Luego se presiona el botón *Generate Source Code to Output*, esto generará el código necesario para la palabra de configuración que se puede copiar y pegar en la ventana del código fuente. Lo anterior se puede observar en la figura 0.13.

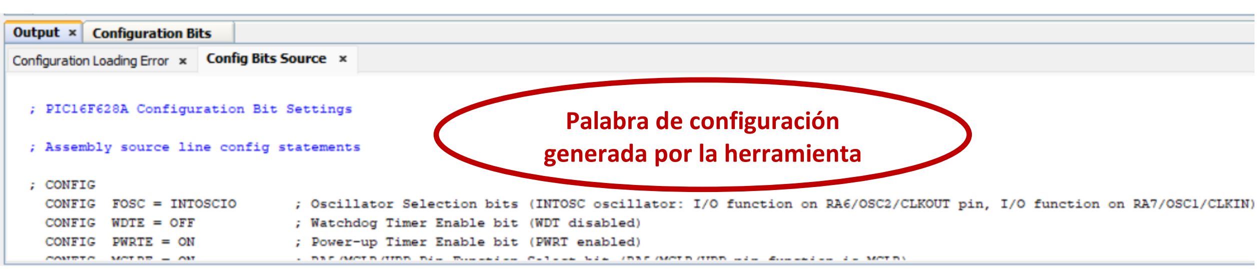


Figura 0.13. Código correspondiente a la palabra de configuración.

Solo queda copiar el código generado de manera automática y pegarlo en la ventana del código fuente como se ve en la figura 0.14.

```
PROCESSOR   16F628A              ;Modelo del microcontrolador

; CONFIG
  CONFIG  FOSC = INTOSCIO        ; Oscillator Selection bits (INTOSC oscillator:
  CONFIG  WDTE = OFF             ; Watchdog Timer Enable bit (WDT disabled)
  CONFIG  PWRTE = ON             ; Power-up Timer Enable bit (PWRT enabled)
  CONFIG  MCLRE = ON             ; RA5/MCLR/VPP Pin Function Select bit (RA5/MCLR
  CONFIG  BOREN = ON             ; Brown-out Detect Enable bit (BOD enabled)
  CONFIG  LVP = OFF              ; Low-Voltage Programming Enable bit (RB4/PGM pi
  CONFIG  CPD = OFF              ; Data EE Memory Code Protection bit (Data memor
  CONFIG  CP = OFF               ; Flash Program Memory Code Protection bit (Code

// config statements should precede project file includes.
#include <xc.inc>
```

Figura 0.14. Palabra de configuración en el código fuente del programa.

El programa que se va a usar como ejemplo posee el siguiente funcionamiento:

La sección llamada "Programa principal" consta de tres partes, la primera llamada REVISA se encarga de hacer una revisión del estado de los botones de "Entrada de Dato" y "Salida de Dato" y dependiendo del botón que se presione salta a una de las otras dos partes.

Si se presionó el botón de entrada, entonces salta a la segunda parte, ENTRA, donde lee el puerto A, aplica el procedimiento llamado *máscara* (pregunte a su profesor para más detalles) y almacena la información dentro de la memoria del microcontrolador en el registro llamado DATO que se declara al principio del programa, cuando termina el proceso regresa a la primera parte de REVISA.

Si se presionó el botón de salida, entonces salta a la tercera parte llamada SALE, donde lee el contenido del registro de memoria DATO y lo escribe en el puerto B para que salga del microcontrolador. Al terminar también regresa a REVISA para comenzar todo el ciclo nuevamente.

Transcriba el programa mostrado en la figura 0.15 al proyecto creado en MPLAB X IDE teniendo cuidado de respetar la sintaxis, recuerde que los comentarios, marcados en color verde, no son parte del programa, solo son una explicación para que el usuario pueda comprender más fácilmente el código del programa.

```
PROCESSOR  16F628A          ;Modelo del microcontrolador

; CONFIG
 CONFIG  FOSC = INTOSCIO    ;Oscillator Selection bits (INTOSC oscillator: I/O function on
                            ;RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN)
 CONFIG  WDTE = OFF         ;Watchdog Timer Enable bit (WDT disabled)
 CONFIG  PWRTE = ON         ;Power-up Timer Enable bit (PWRT enabled)
 CONFIG  MCLRE = ON         ;RA5/MCLR/VPP Pin Function Select bit (RA5/MCLR/VPP
                            ;pin function is MCLR)
 CONFIG  BOREN = ON         ;Brown-out Detect Enable bit (BOD enabled)
 CONFIG  LVP = OFF          ;Low-Voltage Programming Enable (RB4/PGM pin has digital
                            ;I/O function, HV on MCLR must be used for programming)
```

Figura 0.15(a). Código del programa de prueba.

```
 CONFIG  CPD = OFF            ;Data EE Memory Code Protection bit (Data memory code
                              ;protection off)
 CONFIG  CP = OFF             ;Flash Program Memory Code Protection(Code protection off)

// config statements should precede project file includes.
#include <xc.inc>

psect   resetvector,abs,class=CODE,delta=2 ; PIC10/12/16

DATO   EQU 0x20               ;Crea una variable en la localidad 20 hex de la memoria.

;---------- Sección de vectores del programa ---------;
   ORG          0             ;Dirección de memoria del VECTOR DE RESET.
   goto         EMPIEZA
   ORG          5             ;Dirección del INICIO DE LA MEMORIA DE PROGRAMA.

;--------- Sección de configuración de los puertos ----------;
EMPIEZA:      bsf     STATUS,5      ;Cambia al banco 1 de memoria del microcontrolador,
              bcf     STATUS,6      ;<RP1:RP0>=01.

              movlw 0x3F            ;Carga a W con '0011 1111'.
              movwf TRISA           ;Configura a PORTA como entrada.
              movlw 0xF0            ;Carga a W con '1111 0000'.
              clrf    TRISB         ;Configura a PORTB como salida.

;Con lo anterior el PORTA funciona como entrada además de configurar la
;también como entrada la señal del "Botón de entrada de dato" y el MCLR.
;El PORTB se configura como salida y el "Botón de salida de dato" como entrada.

              bcf     STATUS,5      ;Cambia al banco 0 de memoria del
              bcf     STATUS,6      ;microcontrolador, <RP1:RP0>=00.
              movlw 0x07            ;Apaga las funciones analógicas y los
              movwf CMCON           ;puertos entran en modo digital.
              clrf    PORTA         ;Limpia ambos puertos para comenzar el
              clrf    PORTB         ;programa principal.

;--------- Programa principal ----------;
REVISA:       btfsc   PORTA,5       ;Realiza un ciclo donde revisa si se ha presionado
              goto    ENTRA         ;el botón para ingresar un dato o el botón para
              btfsc   PORTB,5       ;extraer un dato y salta a la rutina correspondiente.
              goto    SALE
              goto    REVISA

ENTRA:        movf    PORTA,W       ;Lee el Puerto A y guarda el dato en W.
              andlw  0x0F           ;Aplica una "máscara" a W.
              movwf DATO            ;Se guarda en la variable DATO.
              goto    REVISA        ;Regresa al ciclo REVISA.
```

Figura 0.15(b). Código del programa de prueba.

```
SALE:       movf    DATO,W      ;Lee el contenido de DATO y lo copia en W.
            movwf   PORTB       ;Escribe en el Puerto C el DATO.
            goto    REVISA      ;Regresa al ciclo REVISA.

        END                     ;Fin del programa.
```

Figura 0.15(c). Código del programa de prueba.

Una vez terminada la redacción del código se puede realizar la compilación de este por medio de la herramienta con el ícono de martillo llamada **Build for Debugging** señalada en la figura 0.16. En la parte baja de la ventana aparecerá un mensaje en color verde si el archivo se compiló de manera exitosa o, en caso de existir errores, aparecerán los mensajes de error correspondientes y la línea de código en la que se presentan.

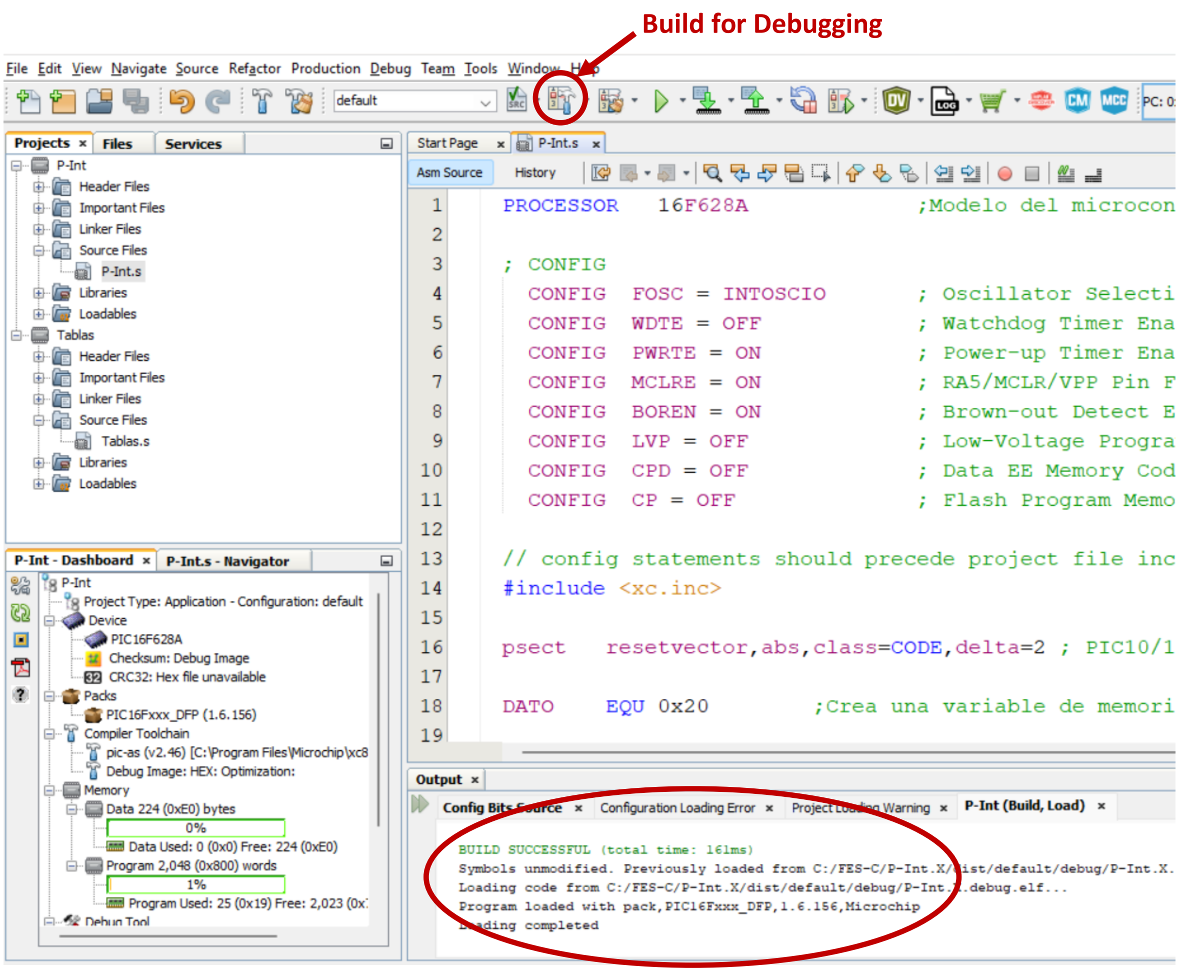


Figura 0.16. Herramienta de compilación y código compilado de manera exitosa.

Una vez que se completó el proceso de escritura y compilación del código se genera dentro de la carpeta del proyecto un archivo con extensión ***.hex***, que es el que se emplea para realizar la simulación del programa y que, posteriormente, se descargará al microcontrolador mediante un dispositivo de programación. En la figura 0.17 se puede observar la carpeta que contiene el archivo ***P-Int.X.production.hex***.

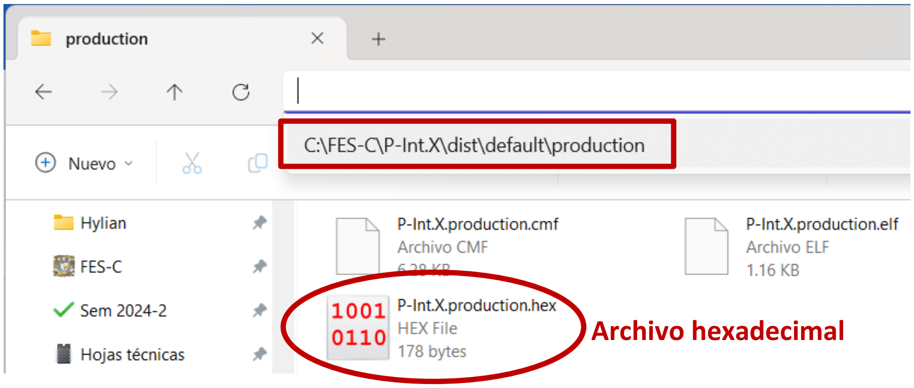


Figura 0.17. Archivo con extensión ***.hex*** resultante de la compilación del programa.

Simulación del programa

Para poder realizar la simulación del programa solo se requiere armar el circuito mostrado en la figura 0.18 dentro del software de simulación empleando los elementos indicados en el lado izquierdo de la figura.

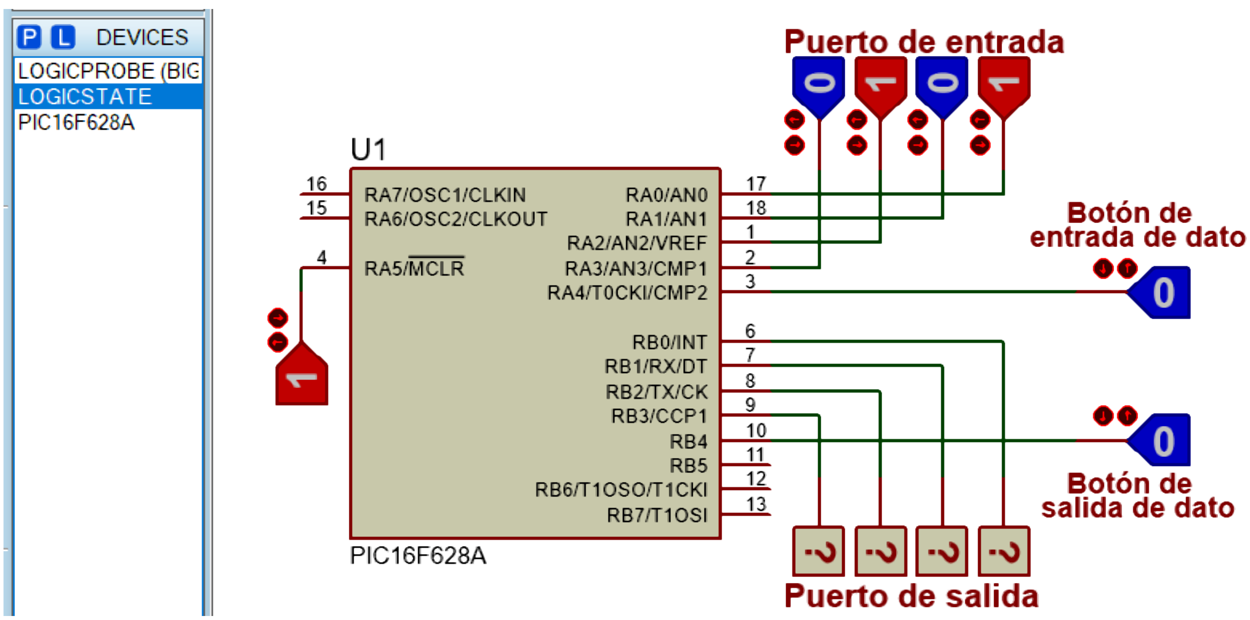


Figura 0.18. Armado del circuito de prueba para el programa.

A continuación, se hace doble clic sobre el microcontrolador para abrir la ventana de propiedades llamada *Edit Component* donde aparecen dos campos de interés que son *Program File* y *Processor Clock Frecuency*, como se muestra en la figura 0.19.

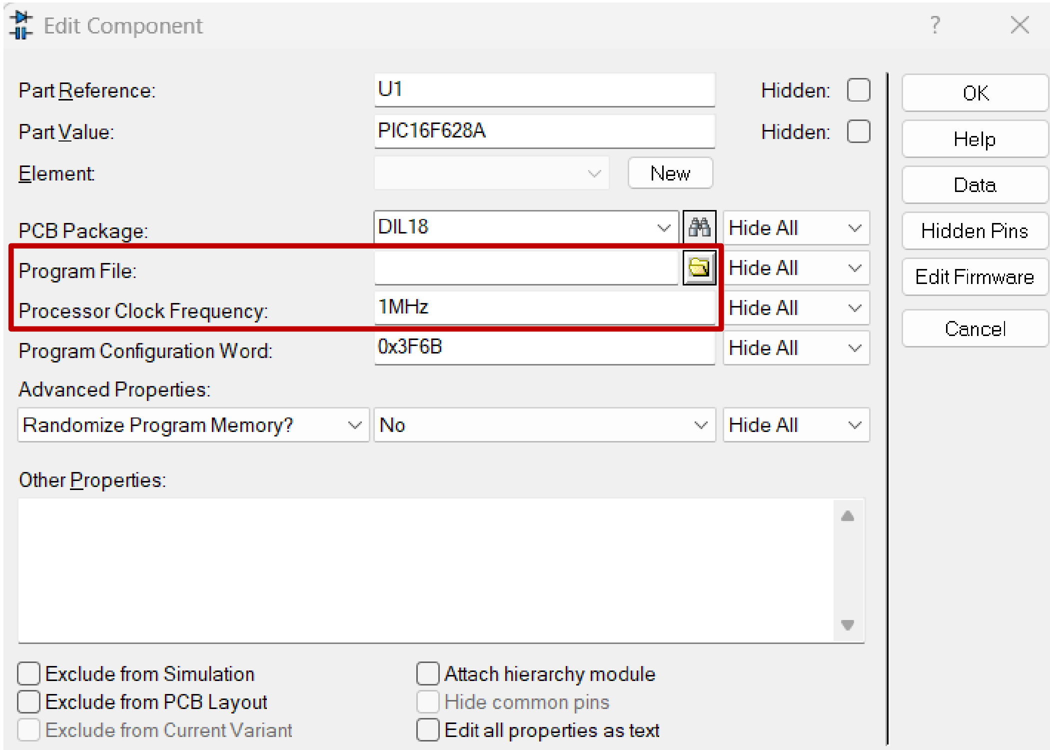


Figura 0.19. Ventana de propiedades del microcontrolador (Edit Component).

Usando el ícono de la carpeta de color amarillo se va a indicar al simulador la ubicación y el nombre del archivo hexadecimal que fue el resultado de la compilación del programa en MPLAB X IDE que en este ejemplo se llama ***P-Int.X.production.hex***. También se va a indicar que la velocidad del reloj del procesador es de 4Mhz y se hace clic en el botón *OK*, como se ve en la figura 0.20.

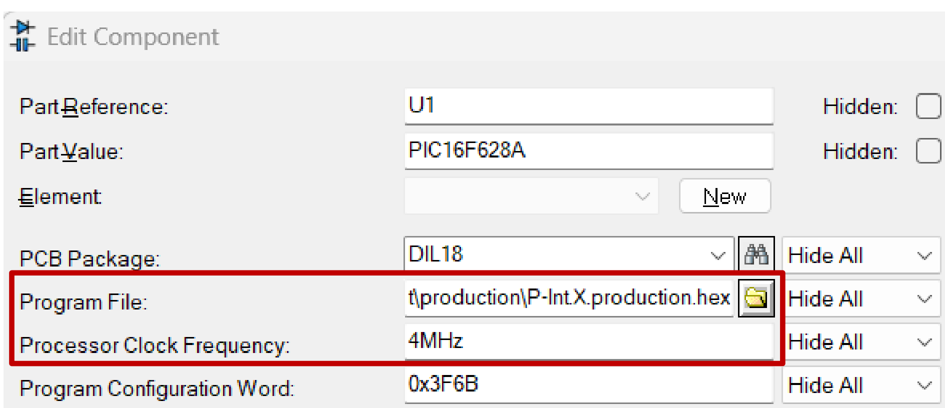


Figura 0.20. Ajustes en la ventana de propiedades del microcontrolador.

Con lo anterior ya es posible comenzar la simulación del circuito con el programa, solo es necesario hacer clic en el botón de ***Play*** ▶ en la esquina inferior izquierda de la ventana y el simulador debe comenzar a funcionar, esto se observa porque los signos de interrogación en las salidas deben cambiar a un valor de cero en color azul, como se ilustra en la figura 0.21.

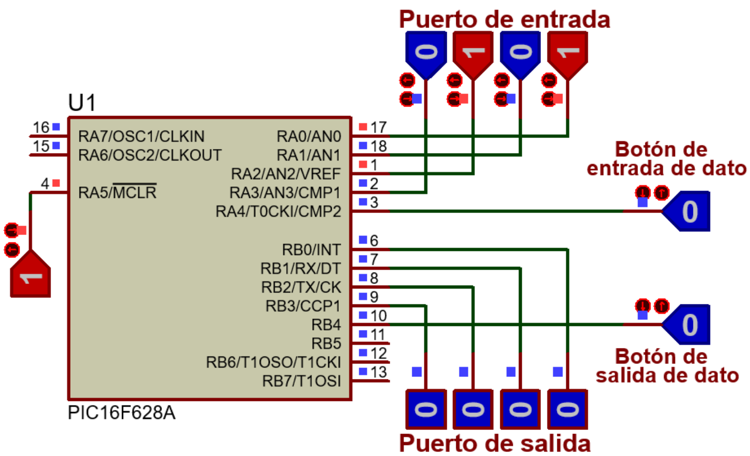


Figura 0.21. Simulación activa.

Para finalizar se puede comprobar el funcionamiento del programa, hay que recordar que el programa lee el puerto de entrada solo si recibe la señal del botón de entrada (1 log), después de eso se debe regresar el botón a 0 log. Después el microcontrolador volverá a están en un estado de espero hasta que se le indique la entrada de un nuevo dato o la salida del que ya se encontraba almacenado. Si se presiona el botón de salida (1 log), el dato que se ingresó anteriormente deberá aparecer en el puerto de salida, después de eso se debe regresar el botón de salida a 0 log. La figura 0.22 muestra en secuencia lo que debe de ocurrir siguiendo la descripción anterior.

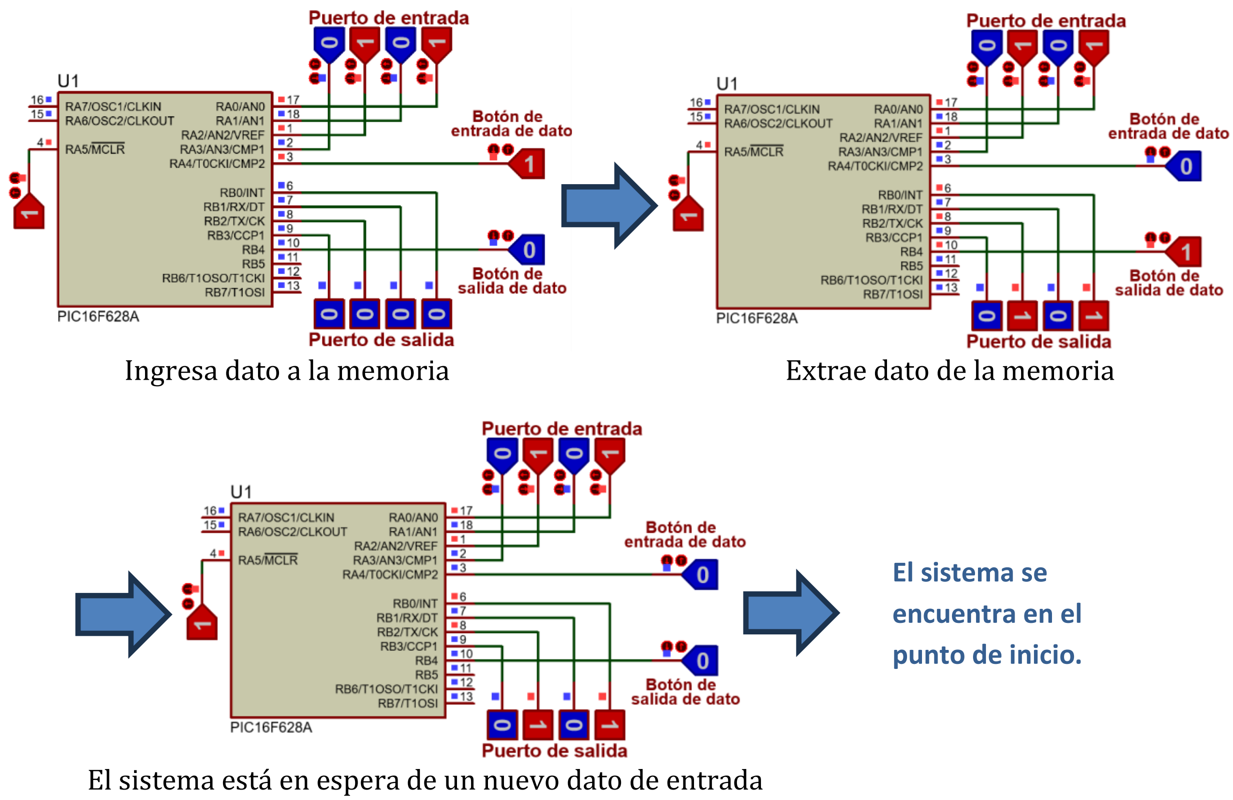


Figura 0.22. Secuencia de funcionamiento del sistema.

Laboratorio de Microcontroladores
Práctica 1
Manejo de señales multiplexadas

Tema

4.1. Multiplexación de señales.

Objetivos

Al término de esta práctica el alumno comprenderá:

- El uso de circuitos multiplexores y demultiplexores.
- La multiplicidad de funciones con que cuenta cada una de las terminales de un microcontrolador y la importancia de su correcta configuración.

Introducción

Los microcontroladores, debido a la gran variedad de opciones especiales y módulos periféricos que pueden llegar a contener, requerirían de una gran cantidad de terminales dedicadas a cada una de estas funciones, lo cual los volvería imprácticos, de gran tamaño y costosos. Es por eso que es necesaria la multiplexación de funciones en cada una de las terminales del microcontrolador. En la figura 1.1 se muestran ejemplos de microcontroladores cuyas terminales están multiplexadas con distintos funcionamientos tanto de entrada como de salida.

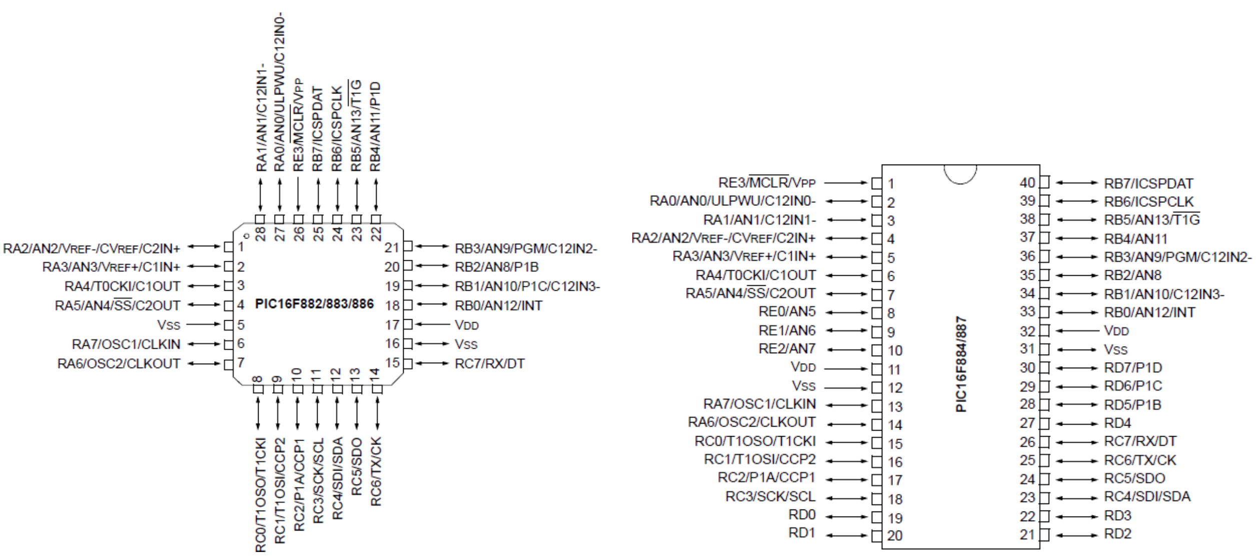


Figura 1.1. Microcontroladores con terminales multiplexadas.

La multiplexación es la combinación de dos o más canales de información en un solo medio de transmisión. En el caso de los microcontroladores, donde pueden configurarse las terminales para actuar como salidas o como entradas, también existe el proceso contrario, llamado demultiplexación donde, usando un solo medio de transmisión, se separan varios canales de información. La figura 1.2 muestra un ejemplo de esto. La terminal o pin de la figura 1.2 del microcontrolador posee funciones digitales de I/O pero adicionalmente esta multiplexada con funciones analógicas de entrada como son el comparador de voltaje y el convertidor analógico digital.

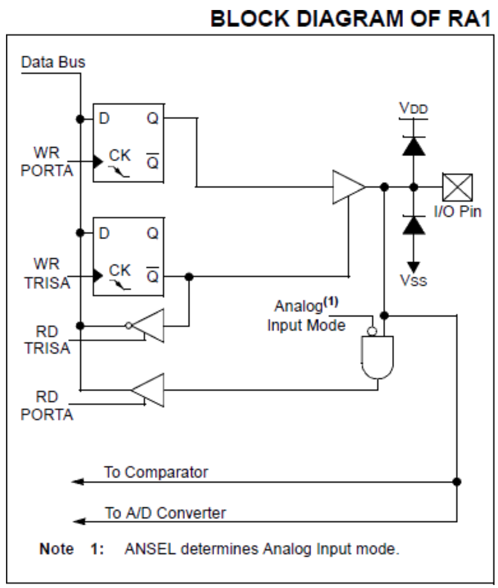


Figura 1.2. Terminal multiplexada de un microcontrolador.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

2) Investigue y tenga a la mano, ya sea en formato electrónico o impreso, la hoja técnica del microcontrolador solicitado en la práctica teniendo especial interés en el diagrama de terminales y las funciones de cada pin de puerto. Muéstresela a su profesor(a) al inicio de la práctica.

3) En el software MPLAB X IDE, escriba el código ensamblador mostrado en la figura 1.3, compile el código y corrija los posibles errores de sintaxis que aparezcan.

4) Realice la simulación del circuito de la figura 1.4 empleando el archivo ***.hex*** obtenido de la compilación del programa de las figuras 1.3 (a) y (b). Compruebe que se comporta de acuerdo con la descripción del punto 2 del procedimiento experimental. Entregue una copia a su profesor siguiendo las indicaciones que éste haya dado para su entrega.

5) **Traer el circuito de la práctica previamente armado**.

Equipo

1 PC con software instalado:

- MPLAB X IDE

1 Grabador universal o grabador de PICs
1 Fuente de voltaje bipolar de CD

Material

1 Microcontrolador PIC12F615
2 Resistencias de 1 kΩ, ½ watt
2 Resistencias de 0.33 kΩ, ½ watt
2 LED de cualquier color
2 Botones tipo push
Tableta de Conexiones
Alambres y cables para conexiones

Procedimiento experimental

1. Escriba y compile el código mostrado en lenguaje ensamblador de las figuras 1.3 (a) y (b).

```
PROCESSOR 12F615      ;Modelo del microcontrolador

;*** Palabra de configuración ***;
    CONFIG FOSC=INTOSCIO   ;Oscilador interno.
    CONFIG WDTE=OFF        ;WDT deshabilitado.
    CONFIG PWRTE=ON        ;PWRT habilitado.
    CONFIG MCLRE=ON        ;MasterClear activado.
    CONFIG CP=OFF          ;Memoria de Programa sin protección.
    CONFIG IOSCFS=4MHz     ;Frecuencia del oscilador interno.
    CONFIG BOREN=ON        ;Brown-out Reset habilitado.

#include <xc.inc>

psect   resetvector,abs,class=CODE,delta=2 ; PIC10/12/16


;*** Sección de vectores ***;
     ORG   0               ;VECTOR DE RESET.
     goto  START
     ORG   5               ;INICIO DE LA MEMORIA DE PROGRAMA.

START:     bsf    STATUS,5        ;Cambio al Banco 1 de memoria.
           movlw  0x04            ;Configura los pines de puerto
           movwf  TRISIO          ;0000 0100.
           clrf   ANSEL           ;Deshabilita las funciones
                                  ;analógicas.
           bsf    STATUS,5        ;Configura al microcontrolador
           movlw  0xA0            ;para trabajar como contador
           movwf  OPTION_REG      ;usando al registro OPTION_REG.
           bcf    STATUS,5        ;Cambio al Banco 0 de memoria.
           clrf   GPIO            ;Limpia el Puerto.
```

Figura 1.3(a). Programa para el contador binario de 2 bits.

```
;*** Programa principal ***;
LOOP:       movf   TMR0,w     ;Se encarga de realizar el conteo y
            andlw  0x03       ;enviarlo al puerto de salida.
            movwf  GPIO
            goto   LOOP
    END                       ;Fin del programa
```

Figura 1.3(b). Programa para el contador binario de 2 bits.

2. El código mostrado en la figura 1.3 permite que el microcontrolador entre en modo de funcionamiento de contador binario, en este caso de dos bits. Cada vez que el circuito recibe dos señales de entrada (pulsaciones) en la terminal GP2 el conteo mostrado en los pines de puerto de salida se incrementa en 1.

3. Programe el microcontrolador con el archivo con extensión ***.hex*** obtenido como resultado de la compilación del programa.

4. Arme el circuito mostrado en la figura 1.4 alimentando el circuito en las terminales $\mathbf{V_{DD}}$ y $\mathbf{V_{SS}}$ de acuerdo con lo que indica la hoja técnica del microcontrolador.

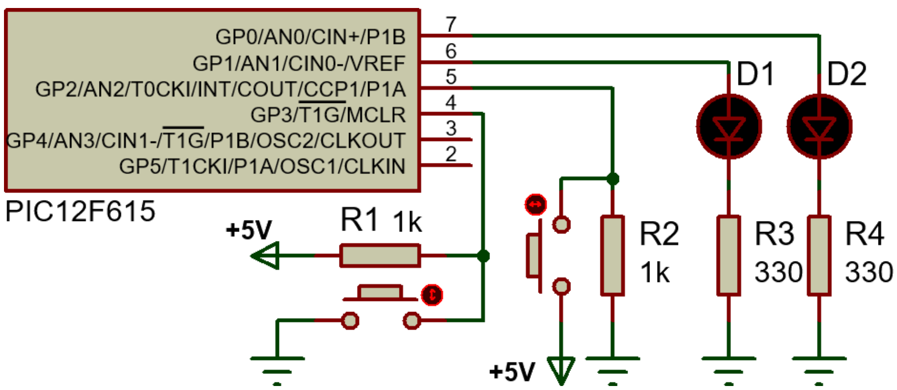


Figura 1.4. Circuito contador binario de 2 bits.

5. **Note que, durante la ejecución de esta función, la terminal 5 del microcontrolador toma una sola de las varias funciones que posee, esto indica que las terminales en el microcontrolador están multiplexadas**.

6. Compruebe el funcionamiento del circuito e incluya fotografías en el reporte de la práctica.

7. A continuación, en el programa de la figura 1.3, comente la línea que apaga las funciones analógicas. Compile nuevamente el programa y descárguelo en el microcontrolador.

8. Pruebe nuevamente el circuito de la figura 1.4 con la modificación en el programa, anotando sus observaciones y comente sobre las razones del comportamiento que presenta el circuito.

9. Modifique el circuito anterior, para obtener el que se muestra en la figura 1.5.

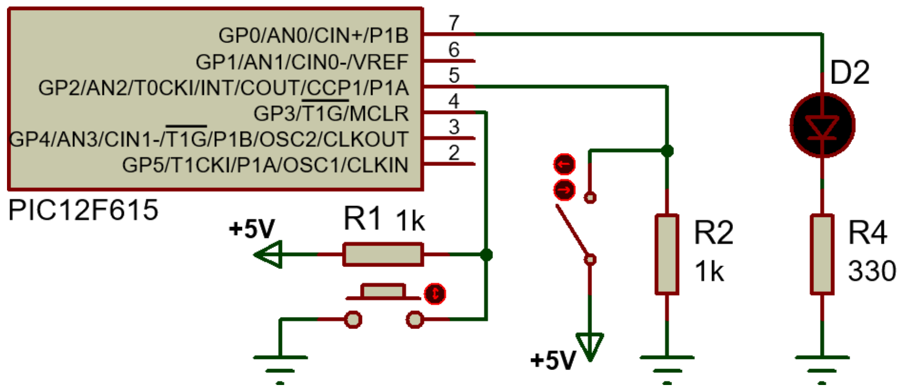


Figura 1.5. Modificación del circuito para uso de pines de puerto.

10. Escriba un nuevo programa en donde se tenga como un pin de entrada digital a GP2 y como pin de salida digital a GP0, esto representa otra de las funciones que pueden tomar las terminales del microcontrolador. El funcionamiento del circuito es el siguiente, siempre que en el pin de entrada exista un 0Log, en el pin de salida también deberá aparecer un 0Log, si en el pin de entrada aparece un 1Log, en el pin de salida deberá observarse también un 1Log.

11. Incluya fotografías del funcionamiento de este último programa en su reporte.

Cuestionario.

1) Explique la diferencia entre un multiplexor y un demultiplexor y porqué es necesario aplicar esta técnica en los circuitos microcontroladores.

2) Para el primer programa de la práctica, indique cuál de las funciones del pin GP2 es la que esta activa y explique brevemente dicha función.

3) Explique la razón por la que es necesario dar dos pulsaciones para que el conteo se incremente en uno en el primer programa de la práctica.

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Laboratorio de Microcontroladores
Práctica 2
Puertos del microcontrolador

Tema

4.2. Descripción funcional de terminales.

Objetivos

Al término de esta práctica el alumno podrá:

- Manipular los puertos del microcontrolador para introducir y extraer datos de este mediante lenguaje ensamblador.
- Configurar los diversos puertos del microcontrolador para que funcionen como puertos de entrada o puertos de salida.
- Utilizar el software MPLAB X IDE de Microchip para compilar programas para microcontroladores.

Introducción

Los microprocesadores PIC tienen diversos puertos de entrada/salida paralelos de usos generales denominados Puerto A, Puerto B, Puerto C, etc. El número de puertos depende del dispositivo que se tenga. Por ejemplo, el PIC16F84A solo tiene dos puertos A y B; mientras que el PIC 16F887 tiene 5 puertos A, B, C, D y E. Para hacer una analogía, los puertos del microcontrolador son similares al ya desaparecido puerto paralelo de la PC, en donde la información entra y sale a través de 8 líneas independientes de datos.

Los puertos del microcontrolador PIC son el medio de comunicación con el mundo exterior, en ellos se pueden conectar los periféricos o circuitos necesarios como por ejemplo los módulos LCD, motores eléctricos, etc., pero estas conexiones no se pueden realizar arbitrariamente. Existen unas reglas básicas que deberán cumplirse para que el microcontrolador no sufra daños o se destruya. Para ello es necesario conocer los límites de corriente que puede manejar el microcontrolador.

Al realizar un programa en lenguaje ensamblador, que será compilado y cargado al microcontrolador PIC, es necesario emplear una serie de directivas para especificar al programa algunos aspectos del microcontrolador que se está empleando, el punto de inicio y fin del programa, etc. A continuación, se describen las directivas más importantes y comúnmente empleadas.

- ***PROCESSOR* <dispositivo>**

Sirve para indicarle al compilador MPLAB X y a su herramienta de compilación *pic-as v2.xx* el modelo de microcontrolador con el que se va a trabajar, lo cual especifica parámetros como el tipo de procesador, opciones de seguridad, opciones de reset, etc., por ejemplo:

PROCESSOR PIC16F6XX donde las **x** representan el modelo particular del microcontrolador

- ***BITS DE CONFIGURACIÓN***

Una vez definido el modelo del microcontrolador y su tipo de procesador, se eligen cuáles de las opciones disponibles serán habilitadas y cuales no, el tipo de oscilador deseado, las protecciones que estarán activas, las opciones de seguridad que se requieran en cada caso, etc. Esto se define mediante etiquetas precedidas por la directiva *CONFIG*, por ejemplo:

CONFIG FOSC = INTRC_NOCLKOUT	Selección de oscilador interno.
CONFIG MCLRE = ON	Habilitación de la terminal *master clear* (MCRL).
CONFIG CPD = OFF	Protección de la memoria de programa deshabilitada.

- ***psect <parametros>***

Es una directiva que define aspectos de la memoria del microcontrolador, como por ejemplo, la ubicación del vector de reset y el inicio de la memoria de programa, los tipos de variables, la longitud de palabra del procesador, etc.

psect barfunc,local,class=CODE,delta=2 ; PIC10/12/16

- ***EQU (<identificador> EQU <exp>)***

Permite crear un registro de propósito general dentro de la memoria de programa, para tal fin se asigna el valor del parámetro <exp>, que representa la dirección de memoria deseada, al identificador. En general, el identificador es un nombre que le es familiar al usuario, por ejemplo:

dato EQU 0x20

Asigna a la dirección 0x20 de la memoria el identificador "*dato*", con esto se puede considerar que dicha dirección funcionará como un registro de propósito general para el programa.

- ***END***

Es una palabra reservada de uso obligatorio y siempre se coloca al final del programa para marcar su finalización. El compilador MPLAB X IDE solo reconoce las líneas de instrucciones que estén escritas previas a la aparición de la directiva END.

Actividades previas a la práctica

1) Realizar la lectura de la práctica.

2) En general, la operación de configuración de los puertos implica la siguiente secuencia:
    - Ingresar al banco 1
    - Configurar los puertos (registros TRISA, TRISB, etc.).
    - Regresar al banco 0
    - Escribir o leer datos desde los puertos (registros PORTA, PORTB, etc.).

3) Desarrolle un programa que configure las líneas de un puerto cualquiera "X" como entrada y las líneas de otro puerto "Y" como salida. El dato de entrada a través del Puerto X será dado en formato **8 4-2-1** de 4 bits mientras que en el Puerto Y será mostrado en formato **Aiken**. Para realizar el programa siga los pasos que se describen a continuación.

Nota: los puertos del microcontrolador no están especificados, cuando se menciona al "Puerto X" se habla de un puerto cualquiera, seleccione el puerto con el que desee trabajar y justifique sus razones.

Nota: recuerde que algunos puertos tienen funciones analógicas que deberán ser deshabilitadas para hacer uso de dicho puerto, revise la hoja técnica para más información.

4) Si se desarrolla el algoritmo se reduce a:
   a) Configurar PX como entrada y PY como salida
   b) W= PX
   c) Convertir dato leído del puerto a formato **Aiken**
   d) PY=W
   e) Ir a paso b)

5) El diagrama de flujo que indica cómo se va a desarrollar el programa es como el mostrado en la figura 2.1.

6) Realice la simulación del sistema de microcontrolador y entregue a su profesor una copia impresa o en formato digital de los resultados obtenidos, de acuerdo con lo indicado por el profesor, donde se aprecien diferentes valores del código de entrada y sus correspondientes salidas. Si la simulación tiene problemas con la barra de leds, reduzca el valor de R6 a 70Ω.

7) Realice una modificación del código del microcontrolador de modo que, al ingresar un dato por el Puerto X, en un Puerto Z aparezca de manera simultánea el dato en código **Gray**. (El Puerto Y mantiene la función que tenía anteriormente).

8) **El circuito deberá ser armado previamente a la realización de la práctica**.

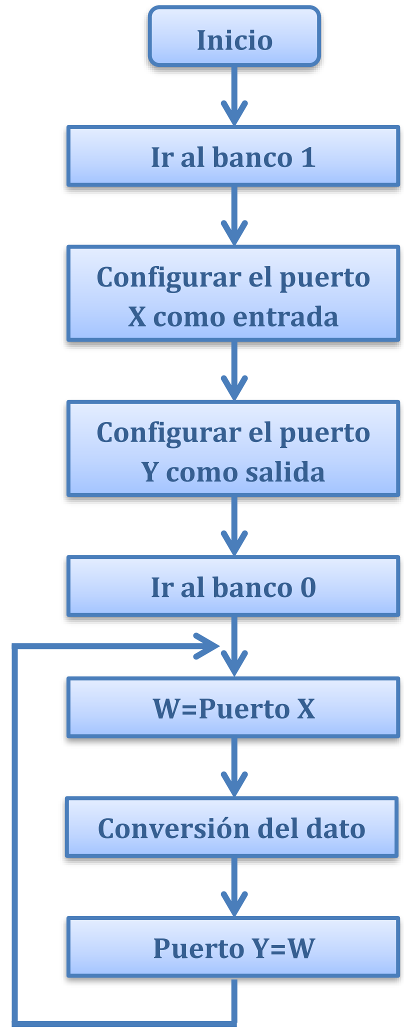


Figura 2.1. Diagrama de flujo del programa de lectura y escritura de puertos.

Equipo

1 PC con software instalado:
- MPLAB X IDE

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD

Material

1 Microcontrolador PIC 16F887
6 Resistencias de 1 kΩ a ½ watt
1 Dip switch de al menos 4 interruptores
1 Push button
1 Barra de leds
Tableta de Conexiones (protoboard)
Alambres y cables para conexiones

Procedimiento experimental

1. Cree un nuevo proyecto en MPLAB X IDE y escriba el programa solicitado en las actividades previas, puede usar como guía el pseudo código mostrado en la figura 2.2 (a) y (b).

```
;***** Programa que lee un dato del puerto X y lo muestra en el puerto Y *****
        PROCESSOR  16F887

;*** CONFIG1 – Primera palabra de configuración ***
        CONFIG  FOSC = INTRC_NOCLKOUT       ; Oscilador Interno.
        CONFIG  WDTE = OFF                  ; Watchdog Timer deshabilitado.
        CONFIG  PWRTE = ON                  ; Power-up Timer habilitado.
        CONFIG  MCLRE = ON                  ; Pin de MCLR active.
        CONFIG  CP = OFF                    ; Protección de la memoria de progr OFF.
        CONFIG  CPD = OFF                   ; Protección de la memoria de datos OFF.
        CONFIG  BOREN = ON                  ; Brown Out Reset habilitado.
        CONFIG  IESO = ON                   ; Bit de Switchover habilitado.
        CONFIG  FCMEN = ON                  ; Monitoreo del reloj.
        CONFIG  LVP = OFF               ; Programación en Bajo Voltaje deshabilitado

;*** CONFIG2 – Segunda palabra de configuración ***
        CONFIG  BOR4V = BOR40V              ; Brown-out Reset activo en 4 volts.
        CONFIG  WRT = OFF                   ; Protección de escritura de memoria Flash.

// config statements should precede project file includes.
#include <xc.inc>

        psect   barfunc,local,class=CODE,delta=2 ; PIC10/12/16
        ; psect   barfunc,local,class=CODE,reloc=2 ; PIC18

;Definición de variables
        dato    equu    0x20            ;Asignar una dirección de memoria a la variable.

;Vectores de reset y de inicio de programa
        org     0
        goto    Inicio                  ;Salta hacia la etiqueta 'Inicio' donde comienza
        org     5                       ;el programa.

;Configuración de puertos
Inicio  bsf     STATUS,5                ;Se posiciona en el banco 1 de la memoria.
        movlw   0xFF                    ;Todos los bits de trisX en uno (1Log) configuran
        movwf   TRISX                   ;al puerto X como entrada.
        movlw   0x00                    ;Todos los bits de trisY en cero (0Log) configuran
        movwf   TRISY                   ;al puerto Y como salida.
        bcf     STATUS,5                ;Se posiciona en el banco 0 de la memoria.
```

Figura 2.2(a). Pseudo Código para la conversión de códigos.

```
;Programa principal
ciclo:  call    leer_puerto_X           ;Llama a la subrutina de lectura.
        call    convertir_dato          ;Llama a la subrutina de conversión.
        call    escribir_puerto_Y       ;Llama a la subrutina de escritura.
        goto    ciclo                   ;Regresa al ciclo principal.

;*** Área de subrutinas ***
;Subrutina para leer el puerto X y guardarlo en dato
leer_puerto_X:
        movf    PORTX,W                 ;Guarda el dato de entrada en el registro W.
        movwf   dato                    ;Envía el contenido del registro W a “dato”.
        return                          ;Regresa de la subrutina.

{ ;Área donde debe escribir la subrutina que convierta el dato de
  entrada al formato de salida }

;Leer el valor de dato y escribirlo en el puerto Y
escribir_puerto_Y:
        movf    dato,W                  ;Envía el contenido de “dato” al registro W.
        movwf   PORTY                   ;Escribe en el puerto Y el código de salida.
        return                          ;Regresa de la subrutina.
                END                     ;Fin del programa.
```

Figura 2.2(b). Pseudo Código para la conversión de códigos.

2. Proceda a armar el circuito en la protoboard de acuerdo con el diagrama de la figura 2.3 y los puertos elegidos al desarrollar el código, programe el microcontrolador y verifique que realice la lectura y escritura de puertos correctamente. Anote sus observaciones.

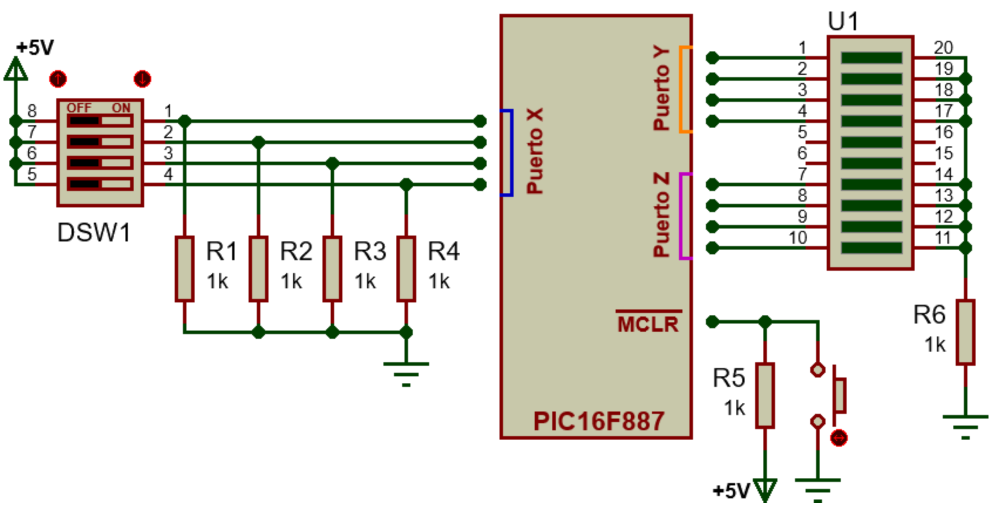


Figura 2.3. Diagrama del circuito.

3. Tome algunas fotografías donde se observe el funcionamiento del circuito para distintos valores en los puertos de entrada y salida.

4. Realice los cambios necesarios al programa principal para que se realice la segunda conversión. Recuerde que el dato a convertir es el del Puerto X, mismo que se convertirá simultáneamente a código **Aiken** (que se mostrará en el Puerto Y) y a código **Gray** (que se mostrará en el Puerto Z). **Recuerde que X, Y y Z hacen referencia a cualquier puerto elegido, no son puertos "reales" del microcontrolador.**

Cuestionario.

1) Dibuje el diagrama de flujo del programa modificado en el punto 4 del procedimiento.

2) ¿Cuáles son los límites máximos de corriente y voltaje totales del PIC16F887 y los que puede soportar un solo pin de puerto? Consulte las características eléctricas en la hoja técnica correspondiente y señálelos.

3) De los distintos puertos con los que cuenta el microcontrolador usado en la práctica, explique las razones para seleccionar los puertos empleados para realizar el programa de la práctica.

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Laboratorio de Microcontroladores
Práctica 3
Temporizadores

Tema

5.3. Programación de microcontroladores.

Objetivos

Al término de esta práctica el alumno podrá:

- Crear retardos a partir del temporizador básico de los microcontroladores PIC, el TIMER 0.

Introducción

Los Temporizadores o “Timers” se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).

Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo, esto lo hace hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Cuando se desean contar acontecimientos que se manifiestan por cambios de nivel o flancos en alguna de las terminales del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

El registro TMR0 es un contador de 8 bits, es decir, un tipo particular de registro cuyo contenido es incrementado con una cadencia regular y programable directamente por el hardware del PIC. Como es de 8 bits, el valor máximo de conteo es 255. Este registro puede usarse para contar eventos externos por medio de un pin de entrada especial (modo contador) o para contar pulsos internos de reloj de frecuencia constante (modo timer). Además, en cualquiera de los dos modos, se puede insertar un prescaler, es decir un divisor de frecuencia programable que puede dividir entre 2, 4, 8, 16, 32, 64, 128 o 256. Este divisor puede ser utilizado alternativamente como prescaler o del TMR0 o como postscaler del Watch Dog Timer, según se le programe.

Una vez alcanzado el valor 255, el registro TMR0 es puesto a cero automáticamente comenzando entonces a contar desde cero y no desde el valor originalmente cargado. La frecuencia de conteo es directamente proporcional a la frecuencia de reloj aplicada al chip y puede ser modificada programando adecuadamente algunos bits de configuración.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

2) Desarrolle un programa que encienda y apague un led de forma intermitente con periodos de 0.45s en cada estado. Puede usar como guía la rutina de tiempo mostrada en la figura 3.1 ajustándola de acuerdo con sus necesidades. **Incluya los cálculos realizados para configurar el módulo del Timer0**.

3) Si se desarrolla el algoritmo se reduce a:
   - Elegir cualquier puerto, por ejemplo, el puerto X y configurar un pin como salida. El pin elegido puede ser cualquiera de los de ese puerto por lo que se le llamará PX.Y.
   - Configurar el tiempo del bit PX.Y en alto.
   - Configurar el tiempo del bit PX.Y en bajo

4) Dibuje el diagrama de flujo del algoritmo anterior.

5) Cree un nuevo proyecto en MPLAB X IDE y empleando las rutinas anteriores, realizar un programa que permita, por medio de dos terminales de selección, multiplicar el periodo de tiempo del programa anterior por 1, por 2, por 3 y por 4.

6) Compile el programa y corrija los posibles errores que surjan.

7) Realice la simulación del sistema de microcontrolador comprobando con ayuda del osciloscopio o del medidor de frecuencia su correcto funcionamiento, guarde la simulación y entregue a su profesor una copia impresa o en formato digital de los resultados obtenidos, de acuerdo con lo indicado por el profesor, donde se aprecien las **mediciones de voltaje, periodo y frecuencia**.

8) Aprovechando las rutinas de tiempo ya realizadas, cree un programa que controle tres pines del Puerto X de modo que funcione como un semáforo con una escala de tiempo corta, considere la secuencia de encendido y que los tiempos no son los mismos para cada una de las tres luces.

9) **El circuito para el primer programa deberá ser armado previamente a la realización de la práctica basándose en el diagrama representativo de la figura 3.2**.

Equipo

1 PC con software instalado:
- MPLAB IDE

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD de 5V
1 Osciloscopio

Material

1 Microcontrolador PIC12F615
1 Resistencia de 1 kΩ, ½ watt
3 Resistencias de 330 Ω, ½ watt
1 Push button
3 Led's (1 Amarillo, 1 Verde y 1 Rojo)
Tableta de Conexiones (Protoboard)
Alambres y cables para conexiones

Procedimiento experimental

1. Haciendo uso del primer proyecto creado anteriormente en las actividades previas, programe el microcontrolador, puede basar el programa en la subrutina de retardo que aparece en la figura 3.1. Recuerde que los valores del número de repeticiones (RR) y el valor de tiempo (TT) mostrados en el código corresponden a los valores calculados en las actividades previas y que PX.Y hace referencia al pin Y del puerto X, es decir, la terminal de salida que se eligió al crear el programa.

```
;Subrutina de retardo
RETARDO:    movlw       0xRR
            movwf       0x20
            call        DELAY
            return
DELAY:      bcf         INTCON,2
            movlw       0xTT
            movwf       TMR0
DELAY_1:    btfss       INTCON,2
            goto        DELAY_1
            decfsz      0x20,f
            goto        DELAY
            return
```

Figura 3.1. Ejemplo de subrutina de retardo para el Timer0.

2. Pruebe el circuito previamente armado de acuerdo con el diagrama representativo de la figura 3.2 y, con ayuda del osciloscopio, grafique la señal de salida del pin PX.Y para el primer programa desarrollado en las actividades previas, haga uso de los cursores del osciloscopio para comprobar que los tiempos en alto y en bajo corresponden con los solicitados en las actividades previas y con la simulación.

3. Tome una fotografía donde se observe el funcionamiento del circuito **indicando el voltaje, periodo y frecuencia de la señal**.

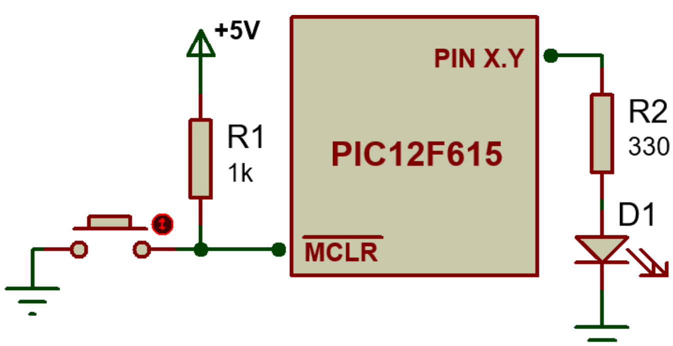


Figura 3.2. Diagrama representativo del circuito del temporizador Timer0.

4. Modifique el circuito anterior para obtener el de la figura 3.3 y programe el microcontrolador con el segundo programa de las actividades previas, de modo que se pueda realizar la prueba que incluye los selectores de la frecuencia de salida.

5. Usando el osciloscopio, observe el funcionamiento del circuito para las distintas combinaciones de los selectores y obtenga las gráficas de funcionamiento **indicando en cada una el voltaje, periodo y frecuencia de la señal**.

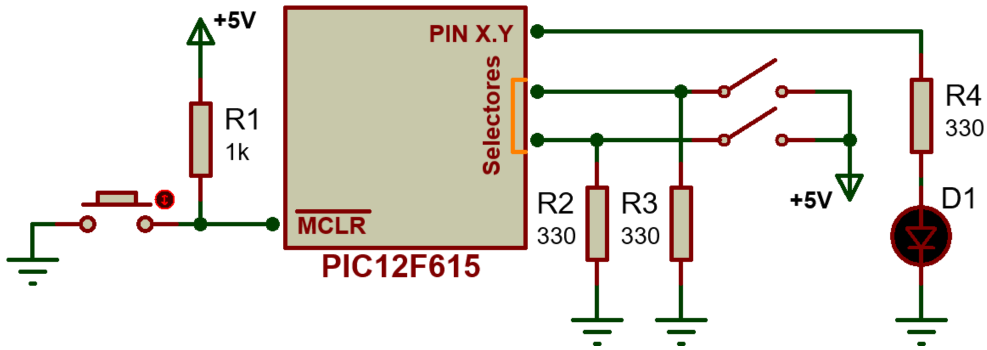


Figura 3.3. Diagrama representativo del circuito temporizador con selectores de frecuencia.

6. Programe el microcontrolador con el tercer programa de las actividades previas y modifique nuevamente el circuito para tener el sistema de semáforo.

7. Compruebe el correcto funcionamiento del semáforo. **Tenga en cuenta que cada luz debe de tener un tiempo de encendido distinto**.

8. Con ayuda del osciloscopio compruebe que los tiempos asignados a cada luz corresponden con los del programa que diseñó y obtenga imágenes donde se observe el voltaje, periodo y frecuencia de la señal en cada luz del semáforo.

Cuestionario

1) ¿Cuál es el valor máximo que puede tener la variable de tiempo "TT" para la subrutina de retardo?

2) Explique el funcionamiento del prescaler del microcontrolador PIC12F675.

3) Determine el tiempo máximo de retardo que se puede lograr con el Timer0 del microcontrolador empleado en la práctica teniendo en cuenta el valor máximo del prescaler y la frecuencia de reloj interna máxima y explique cuáles son sus limitaciones de funcionamiento.

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Laboratorio de Microcontroladores

Práctica 4

Interrupciones

Tema

5.4. Programación de los registros internos.

Objetivos

Al término de esta práctica el alumno podrá:

- Comprender el uso de interrupciones como señales de alta prioridad durante la ejecución de programas.
- Hacer uso del puerto *INT* para detener un proceso y ejecutar una subrutina por medio de la interrupción.

Introducción

Una interrupción es un evento que notifica al CPU del microcontrolador sobre la ocurrencia de una situación excepcional de uno de sus periféricos, es decir, las interrupciones son señales de alta prioridad que permiten que, cuando se produzca un evento interno o externo, se detenga la ejecución del programa principal en cualquier momento. En el momento de producirse la interrupción, el microcontrolador ejecuta un salto al vector de interrupción de la memoria de programa y ejecuta la rutina de atención a la interrupción, previamente definida por el programador, donde se atenderá a la solicitud de la interrupción. Cuando se termina de ejecutar dicha rutina, el PIC retoma a la ejecución del programa principal en el mismo punto donde se produjo la interrupción.

Los microcontroladores PIC de gama baja y media poseen un único vector de interrupción situado en la dirección 04h de la memoria de programa como se ve en la figura 4.1 donde se muestra el PIC12F629 de la gama media, mientras que los de la gama alta tienen dos o tres vectores de interrupción de distinta prioridad, alta y baja respectivamente.

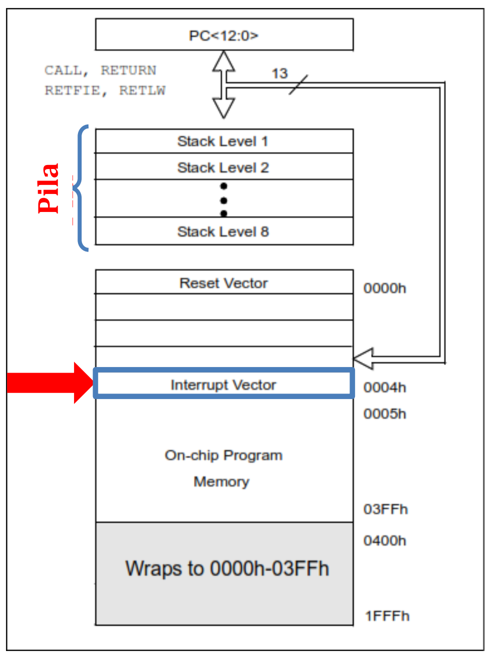


Figura 4.1. Memoria de programa y la pila del PIC12F629.

El número de fuentes de interrupción dependen del microcontrolador utilizado y se dividen en dos tipos: *interrupciones internas*, generadas por los distintos periféricos con que cuenta el microcontrolador; y las *interrupciones externas*, que son detectadas por las terminales del circuito destinadas para esta función. En los microcontroladores PIC de gama media, es común que las terminales con la función de detectar las interrupciones externas se encuentren en el puerto B, más específicamente en el pin RB0 pero esto puede variar por lo que se debe identificar la terminal con la función INT.

Durante la interrupción, la dirección de retomo del programa principal se almacena en la pila y el contador de programa se carga con la dirección 0004h.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

2) Usando el microcontrolador indicado en la lista de material de la práctica, desarrolle un programa en **lenguaje C** que haga que cuatro pines del Puerto X realicen la secuencia mostrada en la figura 4.2, con un intervalo de 0.5s entre cada cambio, de forma cíclica y, por medio de INT, se va a introducir una interrupción que provoque que el puerto completo realice una secuencia de tres parpadeos con un intervalo de 1.5s cada uno. Al finalizar deberá regresar a la rutina inicial justo en el punto donde se interrumpió.

3) El desarrollo del algoritmo se reduce a:
   - Declarar la frecuencia de operación del reloj interno del microcontrolador.
   - Configurar los pines del puerto X como salida, asegurándose que el pin INT es una entrada.
   - Configurar las interrupciones.
   - Crear la rutina de la secuencia de salida de acuerdo con la figura 4.2.
   - Crear la rutina de la interrupción donde se incluya la secuencia de parpadeo.

4) Recuerde que cada vez que se activa la interrupción a través de INT el programa se dirige a la dirección 0004 de la memoria de programa, como se mostró en la figura 4.1. En lenguaje C no es necesario direccionar los vectores de interrupción como ocurre en lenguaje ensamblador, pero **es necesario que la función de interrupción se encuentre antes de la función *main( )***. La función de interrupción se indica como se muestra en la figura 4.3.

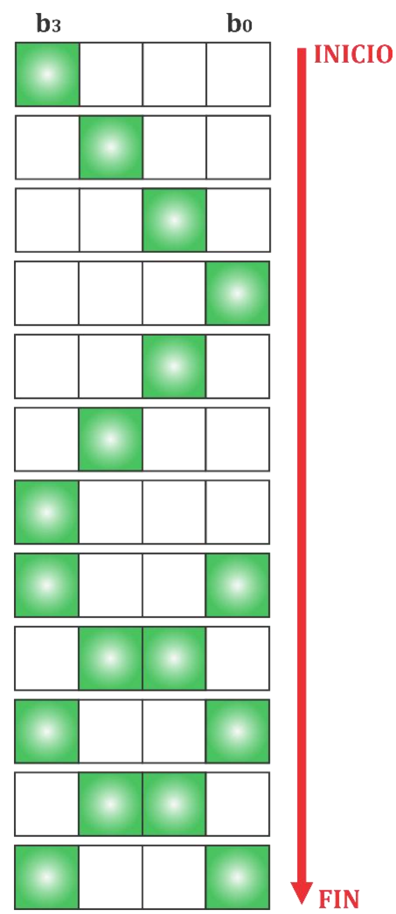


Figura 4.2. Secuencia de funcionamiento del programa principal.

```
Si se requiere declarar alguna variable para el funcionamiento
del programa, se realiza en esta área.

/*Función de interrupción*/
void __interrupt () ISR(void){          //Función correspondiente a la interrupción.
INTCONbits.INTF = 0;                    //Limpieza de la bandera de interrupción.

Área donde se especifican las acciones a realizar
durante la interrupción.

        }
/*Función principal*/
void main( ){                           //La función de interrupción esta antes de main( )
```

Figura 4.3. Ejemplo de la forma para incluir la función de interrupción.

5) Tome en cuenta que, para configurar los pines de puerto, deshabilitar las funciones analógicas y habilitar las interrupciones, se deben configurar los registros necesarios y esto se debe colocar **dentro de la función *main( )***. Las instrucciones para la configuración de las interrupciones se muestran en el ejemplo de la figura 4.4, los ** indican que ahí se deben de colocar los valores de configuración determinados previamente.

```
/* Función principal */
void main( ){

    TRISIO=**;                  //Configura el puerto GPIO.
    GPIO=**;                    //Limpia el puerto.
    ANSEL=**;                   //Deshabilita las funciones analógicas.
    INTCON=**;                  // Configuración de interrupciones.

    // Programa principal – ciclo continuo.
    while(True) {

      Área donde se especifican las acciones que
      realizará el programa principal.

            }                   // Fin del ciclo while.
        }                       // Fin de la función main( ).
```

Figura 4.4. Ejemplo de la forma de configuración de la función de interrupción.

6) Para obtener los tiempos de encendido en las secuencias del programa principal, puede utilizar la directiva de tiempo *__delay_ms( )*.

7) Compile el programa con MPLAB X IDE, corrigiendo los posibles errores.

8) **El circuito deberá ser armado previamente a la realización de la práctica.**

Material

1 Microcontrolador PIC12F615
2 Resistencias de 1 kΩ, ½ watt
4 Resistencias de 330 Ω, ½ watt
2 Push button
4 Leds
Tableta de conexiones
Alambres y cables para conexiones

Equipo

1 PC con software instalado:
- MPLAB X IDE

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD
1 Osciloscopio

Procedimiento experimental

1. Haciendo uso del proyecto creado anteriormente y siguiendo la lógica del algoritmo de las actividades previas, programe el microcontrolador.

2. Con el circuito previamente armado, como el mostrado en la figura 4.5, verifique que realice la secuencia de forma correcta

3. Conecte los dos canales del osciloscopio a dos de las señales de salida del microcontrolador midiendo el tiempo de duración en alto para comprobar que los tiempos programados son correctos. Incluya imágenes de las mediciones en su reporte.

4. Tome fotografías donde se observe el funcionamiento del circuito e inclúyalas junto con las imágenes del osciloscopio en su reporte.

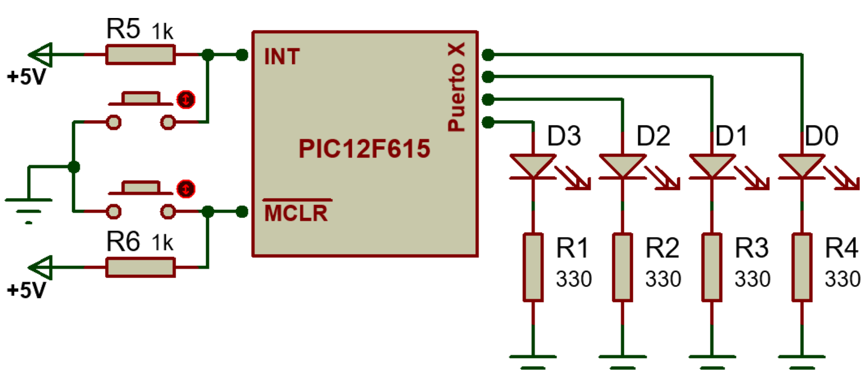


Figura 4.5. Diagrama representativo del circuito secuencial con interrupción externa.

5. Revise la hoja técnica del microcontrolador y busque la forma en que se debe de configurar para que la interrupción INT se active con flancos de bajada, es decir, cuando la terminal INT pase de un estado alto a uno bajo.

6. Modifique el programa anterior para que ahora se tenga la activación de la interrupción en flancos de bajada, programe el microcontrolador con dicha modificación y pruebe se funcionamiento.

7. Describa lo ocurrido de manera física en su circuito al utilizar el programa modificado.

Cuestionario.

1) De manera general, las interrupciones en un microcontrolador pueden dividirse en dos grupos, mencione dichos grupos y las características de cada uno.

2) El microcontrolador PIC16F887 cuenta con numerosas fuentes de interrupción, describa detalladamente al menos tres distintas a la empleada en la práctica. **Revise la hoja técnica.**

3) Investigue en la página del fabricante del microcontrolador usado en la práctica si existen otros modelos que posean más de una fuente de **interrupción externa**. De ser así, de dos ejemplos.

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Laboratorio de Microcontroladores
Práctica 5
Temporizadores y contadores

Tema

6.3. Elaboración de proyectos.

Objetivos

Al término de esta práctica el alumno podrá:

- Crear rutinas de retardo a partir de programación en C para microcontroladores.
- Utilizar uno de los módulos de temporización en modo de contador para identificar eventos que se produzcan de forma externa al microcontrolador.

Introducción

Los temporizadores son módulos periféricos que se encuentran dentro de los microcontroladores, tienen la función de crear retardos de tiempo en base a registros que se incrementan de acuerdo con un oscilador interno asociado a ellos. Típicamente, existen temporizadores de 8 y 16 bits y varía el número de módulos con cada modelo de microcontrolador. Al tratarse de registros que se van incrementando en cada *evento del oscilador*, es decir, con cada pulso de reloj, solo pueden ser programados con valores de tiempo discretos de dicho oscilador, por lo que el microcontrolador cuenta con otro dispositivo auxiliar llamado *prescaler*, el cual actúa como un divisor de frecuencia para poder reducir la velocidad de cada evento. Hay que recordar que, como el registro del temporizador cambia su valor en cada nuevo evento, estos temporizadores son de tipo ascendente.

Hay otras consideraciones que deben de tenerse en cuenta al programar el módulo del temporizador, por ejemplo, el valor que puede tomar el *prescaler*, que se puede consultar de tablas incluidas en las hojas técnicas del microcontrolador, la frecuencia del reloj con la que opera el μc, la fuente del reloj y el lenguaje de programación en que se realiza el código. Es necesario realizar estas consideraciones para poder obtener un resultado apropiado.

Otra forma de lograr retardos de tiempo es mediante software, usando la programación en lenguaje C. Existen directivas de pre-procesado que, al ser incluidas dentro del código de programación, permiten crear estos retardos simplemente especificando la frecuencia del oscilador del microcontrolador y usando instrucciones específicas de retardo dentro del código. Hay que tener en cuenta que, al ser de alto nivel el lenguaje C, no conocemos la forma exacta en la que genera estos retardos, pues varía con cada compilador.

Existe una función adicional asociada a algunos temporizadores que permite usar señales externas como la fuente de reloj del temporizador. Cuando se utiliza esta opción, se dice que el temporizador funciona en *modo de contador*. Con cada evento externo, el valor del registro del temporizador se incrementará en uno. Normalmente se destina para este modo de funcionamiento los temporizadores de 16 bits.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

2) Defina los términos: *periodo, frecuencia, señal de reloj digital, flanco de subida, flanco de bajada, tiempo de respuesta en circuitos digitales, niveles de voltaje TTL* y *efecto de rebote*.

3) Para el correcto armado del circuito es importante la orientación de los elementos optoelectrónicos, investigue en las hojas técnicas los valores del ángulo de dispersión y la longitud de onda del led IR de la práctica. Incluya las gráficas de distribución espectral.

4) Desarrolle un programa que permita que el microcontrolador funcione como una "fuente de reloj digital". El circuito debe contar con dos entradas de selección que permitan cambiar la frecuencia de la señal de salida con los valores de "apagado", 2.2 kHz, 6.4 kHz y 12 kHz. **NOTA**. Considere que, debido al proceso de traducción del lenguaje C a lenguaje máquina y los límites de operación del microcontrolador, es posible que en la prueba real en el laboratorio existan errores dentro de un rango de ±3% en las frecuencias, **esto es normal**.

5) El algoritmo debe seguir la siguiente lógica:
   - Configurar los pines de entrada del puerto PX.Y y PX.Z
   - Configurar el pin de salida del puerto PX.W (NOTA: se está usando un solo puerto)
   - Crear un ciclo que haga una revisión del estado de los pines de entrada PX.Y y PX.Z
   - De acuerdo a la combinación de entrada, configurar los tiempos que permanecerá en alto y en bajo el pin PX.W para la frecuencia seleccionada.
   - En caso de que la combinación corresponda a "apagado", suspender la emisión de señales en el pin PX.W.

6) Realice el diagrama de flujo que indique cómo se va a desarrollar el programa.

7) Compile el programa con MPLAB IDE, corrigiendo los posibles errores.

8) La figura 5.1 muestra señales de baja frecuencia con distintos anchos de pulso, realice un nuevo programa que permita generar dichas señales con las características indicadas. El programa debe funcionar en base a un interruptor para elegir el ancho de la señal.

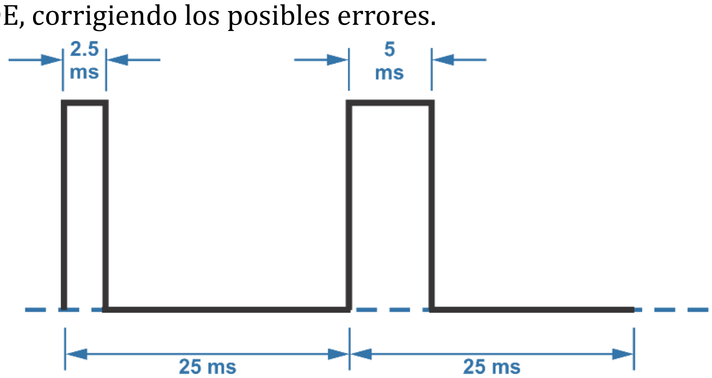


Figura 5.1. Señales de con ancho de pulso variable.

9) Compile el nuevo programa con MPLAB IDE, corrigiendo los posibles errores.

10) Escriba un tercer programa que, usando el temporizador de 16 bits, realice el conteo de una serie de eventos externos al microcontrolador y muestre el valor del conteo en un display por cada evento en una línea y en la otra línea un conteo por cada grupo de 8 eventos.

11) El algoritmo debe tener la siguiente lógica:
    - Incluir en las directivas de pre-procesado la biblioteca de control del display.
    - Configurar como entrada el pin de puerto correspondiente al TMR1.
    - Configurar el display con cuidado de no usar el mismo puerto que el que usa el TMR1
    - Mostrar, en las líneas del display, los mensajes del valor de ambos conteos.
    - Crear un ciclo que haga una revisión del estado del pin de entrada del TMR1.
    - En caso de se presente un flanco de subida en el pin de entrada, el conteo deberá incrementarse en 1 y actualizarse en la primera línea del display y cada 8 eventos se deberá actualizar el valor de la segunda línea de conteo

12) Compile el programa con MPLAB IDE, corrigiendo los posibles errores.

13) Realice la simulación del sistema de microcontrolador para los tres programas, guarde las simulaciones y entregue a su profesor una copia impresa o en formato digital de los resultados obtenidos, de acuerdo con lo indicado por el profesor, donde se aprecien las mediciones de voltaje, periodo y frecuencia para los dos programas de temporizador y ejemplos de distintos valores que indique el contador del tercer programa.

14) **Los circuitos deberán ser armados previamente a la realización de la práctica**. Si elige con cuidado las conexiones del microcontrolador se pueden colocar todos los elementos para los programas solicitados en un mismo circuito sin que haya necesidad de modificarlo posteriormente.

NOTA: Dentro de la simulación, el diodo IR y el fototransistor se puede sustituir por el elemento llamado "optocoupler" y un interruptor para aparentar la obstrucción del rayo de luz infrarroja, como se muestra en la figura 5.2.

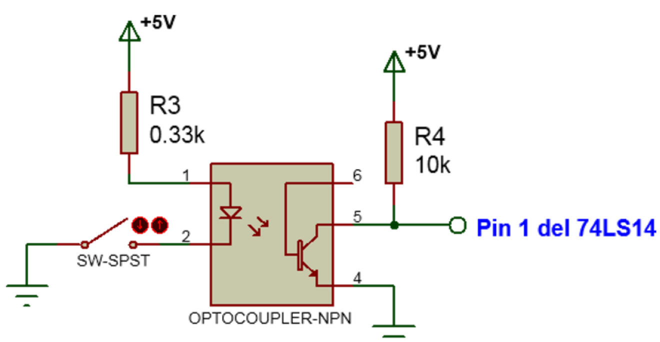


Figura 5.2. Optoacoplador para simulación.

Material

1 Microcontrolador PIC 16F887
1 Display LCD 16x2
1 Potenciómetro de 2 kΩ
4 Resistencias de 10 kΩ, ½ watt
1 Resistencias de 0.33 kΩ, ½ watt
1 Diodo emisor de luz infrarroja, led IR333C.
1 Fototransistor infrarrojo, PT331C
1 Inversor Schmitt Trigger, 74LS14
1 Push button
Tableta de Conexiones (Protoboard)
Alambres y cables para conexiones

Equipo

1 PC con software instalado:
- MPLAB X IDE
- Simulador de circuitos

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD
1 Osciloscopio

Procedimiento experimental

1. Haciendo uso del primer programa de las actividades previas y siguiendo la lógica de su algoritmo programe el microcontrolador.

2. Con ayuda del osciloscopio, en acoplo de CD, verifique que el circuito de la figura 5.3 genere de manera correcta las señales de reloj de salida considerando que, **si la frecuencia no es exacta**, será aceptable mientras esté dentro de un rango de error de ±3%.

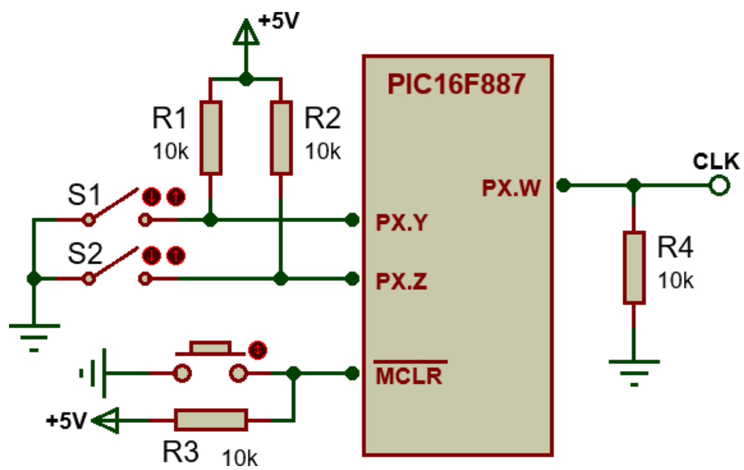


Figura 5.3. Circuito simbólico del sistema de reloj digital.

3. Incluya una imagen donde se observe el funcionamiento del circuito, **acotando cuidadosamente** las características de la señal de reloj como son frecuencia, periodo, tiempo en alto y tiempo en bajo.

4. Cambien la posición del interruptor de selección y observe en el osciloscopio el cambio de frecuencia en la señal de salida.

5. Incluya una imagen donde se observe el cambio en la frecuencia de salida y acótela por completo como se realizó con la señal anterior.

6. Utilizando el mismo circuito del programa anterior, figura 5.3, cargue en el microcontrolador el segundo programa de las actividades previas.

7. Con ayuda del osciloscopio compruebe el correcto funcionamiento del nuevo programa e incluya una imagen de ambos casos donde estén indicadas **todas las acotaciones necesarias**. En caso de existir errores en las frecuencias medidas con el osciloscopio, procure reducirlos hasta donde sea posible, es decir, la mejor aproximación al valor solicitado.

8. A continuación, para el circuito de la figura 5.4, cargue en el microcontrolador el tercer programa de las actividades previas.

   NOTA: Al armar este circuito coloque el fototransistor a una distancia de entre 5 y 10 cm del led infrarrojo.

9. Pruebe que, al interrumpir el haz de luz infrarroja, se incremente el valor del conteo mostrado en la pantalla del display.

10. Aplique una señal de reset y observe el comportamiento del circuito.

11. Incluya fotografías del funcionamiento del circuito de conteo.

12. Coloque uno de los canales del osciloscopio a la salida del fototransistor y el otro a la salida del circuito Schimth Trigger. Interrumpa el haz infrarrojo en varias ocasiones y observe ambas señales. Anote sus observaciones e incluya una imagen.

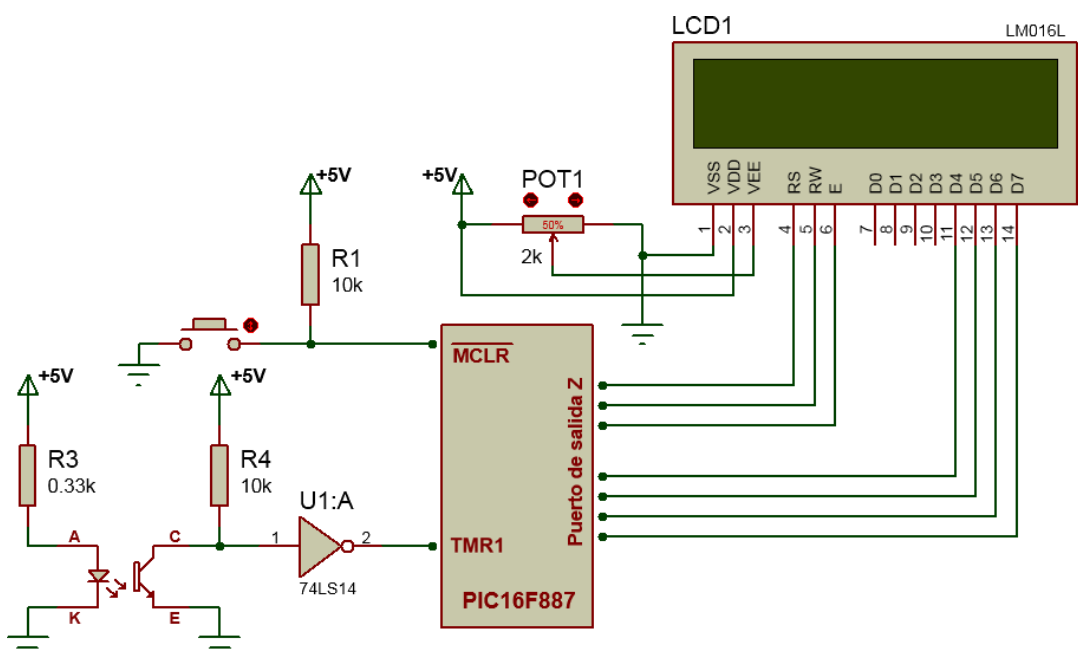


Figura 5.4. Circuito simbólico del sistema de conteo con sensor infrarrojo.

Cuestionario.

1) Explique la diferencia entre generar retardos de tiempo por medio de la directiva *delay* y el uso de alguno de los temporizadores del microcontrolador.

2) Escriba un programa en lenguaje C que, utilizando específicamente el módulo del Timer0, realice el mismo funcionamiento que el primer programa realizado en esta práctica.

3) Si el límite del conteo de un sistema es mayor al que se puede alcanzar con el módulo del Timer1 de la práctica, ¿cómo solucionaría dicha dificultad sin cambiar por otro circuito?

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Tema

7.2. Diseño de interfaces de control de potencia eléctrica.

Objetivos

Al término de esta práctica el alumno podrá:

- Controlar el sentido de giro y posición angular de un motor de pasos utilizando los modos de operación a paso completo y medio paso.

- Implementar un sistema con un motor a pasos de tipo bipolar empleando un control basado en un microcontrolador.

Introducción

Los motores a pasos son dispositivos electromecánicos que convierten una serie de impulsos eléctricos en desplazamientos angulares discretos. Un ejemplo de estos motores se puede observar en la figura 6.1. Estos motores presentan la ventaja de tener alta precisión y repetitividad en cuanto al posicionamiento por lo que son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos.

Figura 6.1. Motor a pasos

La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8°, es decir, que se necesitarán 4 pasos en el primer caso y 200 pasos para el segundo caso, para completar una rotación de 360°. Para mantener la marcha del motor es necesario cambiar periódicamente la combinación de pulsos en sus terminales, como se ilustra en la figura 6.2.

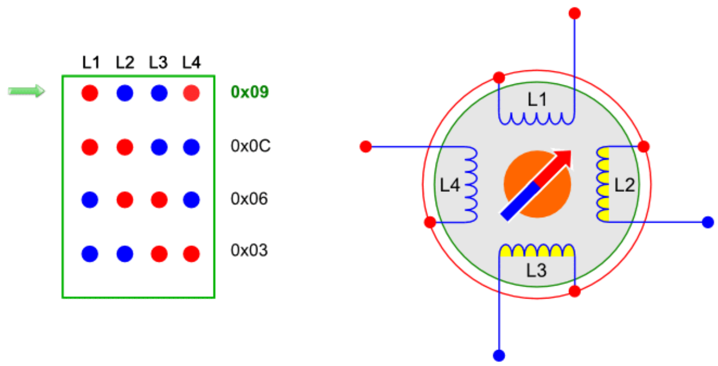


Figura 6.2. Ejemplo de secuencia de activación de un motor a pasos (Full Step).

El ángulo de rotación del eje es directamente proporcional a la secuencia de pulsos insertados a las bobinas y la velocidad de rotación es dependiente de la frecuencia de dichos pulsos. Los motores a pasos son simples de operar en una configuración de lazo abierto y debido a su tamaño proporcionan un excelente torque a baja velocidad.

En combinación con circuitos de control, además del movimiento ilustrado en la figura 6.2, conocido como *paso completo* (full step), es posible lograr movimientos aún más precisos. Este otro modo de operación se conoce como movimiento de *medio paso* (half step). Para lograrlo, se polarizan las bobinas de a una y de a dos intercaladamente, como se muestra en la figura 6.3. Se observa que también incluye los 4 pasos del modo full step. Obviamente esos son los momentos en que hay dos bobinas polarizadas, en los otros 4 pasos, sólo se polariza una bobina. La ventaja de este mecanismo respecto del modo full step es que se pueden realizar movimientos de giro más finos.

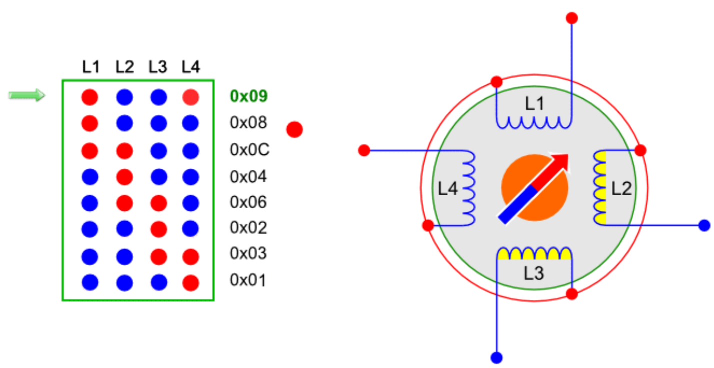


Figura 6.3. Ejemplo de secuencia de activación a medio paso (Half Step)

En los sistemas de control moderno se presentan a menudo movimientos de tipo incremental, por esto los motores de pasos se han convertido en elementos de acción importantes y en la actualidad podemos encontrar estos motores en unidades de discos ópticos, unidades de disco duro, impresoras, en gran variedad de máquinas herramientas además de ser fundamentales para proporcionar movimiento a los robots.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

2) Investigar la forma en que se pueden identificar las terminales de un motor a pasos y la clasificación a la que pertenece.

3) Desarrolle un programa para que el microcontrolador realice el control del sistema de motor a pasos de la figura 6.7 con las siguientes funciones:
   - A través de una señal de control, se selecciona el modo de funcionamiento entre paso completo y medio paso.
   - Por medio de interruptores de selección, el motor debe girar de manera continua hacia la derecha o izquierda, con una frecuencia en las señales de cada fase de 40 Hz.
   - Al presionar alguno de los botones, el motor debe girar hacia la derecha o izquierda exactamente 180° (1/2 vuelta), con una frecuencia en las señales de control de 40 Hz.
   - Siempre que el motor esté activo, en el display deberá mostrase un mensaje que indique el modo de funcionamiento, el sentido de giro y el tipo de paso.

4) Realizar la simulación de todos los puntos del desarrollo experimental usando el archivo con extensión ***.hex*** que obtuvo al compilar el código, guarde la simulación y entregue a su profesor una copia impresa o en formato digital de los resultados obtenidos, de acuerdo con lo indicado por el profesor, donde se aprecien las señales de cada una de las fases del motor, el periodo y la frecuencia, además del funcionamiento del display.

5) El motor a pasos con el que se cuenta en el laboratorio es un **motor bipolar de 200 pasos**, para ajustar en la simulación el número de pasos del motor solo es necesario ingresar a las propiedades del motor e indicar el ángulo por cada paso que se desea que tenga y el voltaje nominal de alimentación. La ventana de propiedades se muestra en la figura 6.4.

6) **El circuito de la figura 6.7 deberá ser armado previamente a la realización de la práctica**.

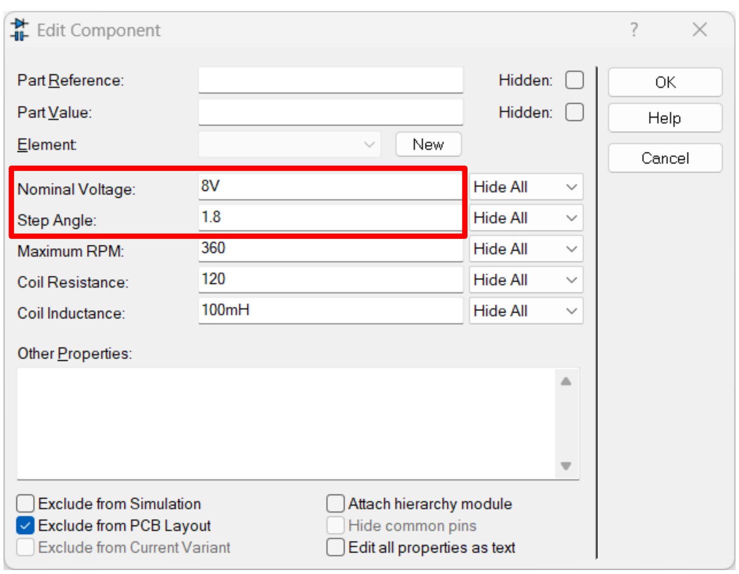


Figura 6.4. Ventana de propiedades del motor a pasos.

Equipo

1 PC con software instalado:
- MPLAB X IDE
- Simulador de circuitos

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD
1 Osciloscopio
1 Multímetro
1 Motor bipolar de 200 pasos

Material

1 Microcontrolador PIC16F887
1Controlador dual de puente completo L298N
1 Pantalla LCD de 16x2
6 Resistencias de 1 KΩ a ½ W
1 Potenciómetro de 2 KΩ
2 Capacitores de 0.1 μF
3 Push Button normalmente abiertos
8 Diodos 1N4004
Tableta de Conexiones (Protoboard)
Alambres y cables para conexiones

Nota: Puede sustituirse el controlador L289N, los diodos 1N4004 y los capacitores de 0.1μF por un módulo integrado como el mostrado en la figura 6.5, **recuerde traer un desarmador**.

Figura 6.5. Módulo de control de motor a pasos.

Procedimiento experimental

1. De acuerdo con lo investigado en las actividades previas, determine la secuencia de las fases del motor a pasos de la práctica.

2. Haciendo uso del programa desarrollado en las actividades previas, y siguiendo la lógica de su algoritmo, programe el microcontrolador.

3. Conecte las salidas de las fases del circuito L298 a las conexiones del motor de pasos, figura 6.6, siguiendo la secuencia determinada en el punto 1 del procedimiento.

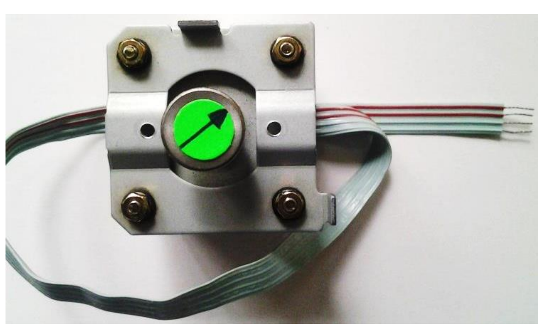


Figura 6.6. Fases del motor bipolar a pasos.

4. Compruebe que, para el circuito mostrado en la figura 6.7, las conexiones correspondan con las terminales de los puertos elegidos al realizar el programa, como entrada (Puerto X), salida (Puerto Y) y para el display (Puerto Z).

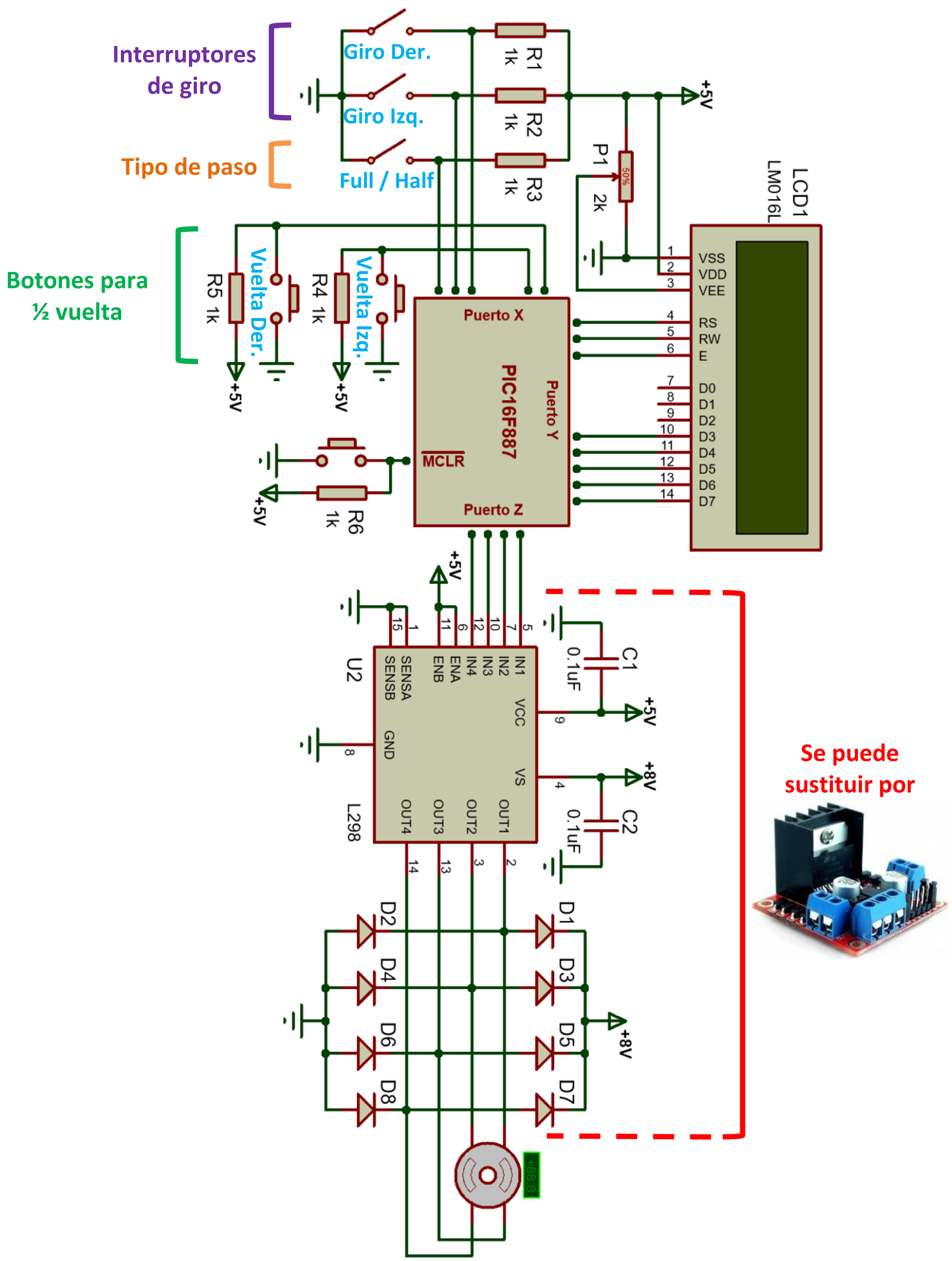


Figura 6.7. Circuito simbólico de control de motor a pasos.

5. Pruebe el correcto funcionamiento del circuito, comenzando por las funciones de giro continuo, tanto para paso completo como medio paso.

6. La tabla 6.1 presenta el funcionamiento que debe de tener el motor de acuerdo con el circuito de control de la figura 6.7, **observe que la tabla usa lógica inversa**.

Tabla 6.1. Lógica de funcionamiento del programa.

Paso	**Botones**		**Interruptores**		
Full/Half	**Izq.**	**Der.**	**Izq.**	**Der.**	**Función**
0	1	1	1	0	Giro continuo a la derecha medio paso
0	1	1	0	1	Giro continuo a la izquierda medio paso
0	1	0	1	1	Vuelta y media a la derecha medio paso
0	0	1	1	1	Vuelta y media a la izquierda medio paso
X	X	X	1	1	Motor detenido
1	1	1	1	0	Giro continuo a la derecha paso completo
1	1	1	0	1	Giro continuo a la izquierda paso completo
1	1	0	1	1	Vuelta y media a la derecha paso completo
1	0	1	1	1	Vuelta y media a la izquierda paso completo

7. Con ayuda de ambos canales del osciloscopio, en acoplo de CD, verifique que el circuito genere de manera correcta las señales de salida en fases consecutivas que van hacia el circuito L298 y mida su frecuencia, periodo. Esto debe comprobarse para ambos modos, paso completo y medio paso.

8. A continuación, compruebe que, al presionar alguno de los botones de medio giro a la derecha o izquierda, el motor realice el movimiento de forma exacta y se detenga. Esto debe comprobarse para ambos modos, paso completo y medio paso.

Cuestionario.

1) ¿Cuáles son las ventajas y desventajas de la operación FULL step con respecto al modo HALF step?

2) Escriba en lenguaje C una subrutina para controlar el movimiento del motor de modo que, cada que reciba un pulso proveniente de un nuevo botón agregado al sistema, la posición se incremente en un solo paso (esto se conoce como Jogging).

3) ¿Qué es la sincronía en motores a pasos y por qué es importante mantenerla?

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Laboratorio de Microcontroladores

Práctica 7

Comparadores de voltaje

Tema

7.5. Control con optoacopladores.

Objetivos

Al término de esta práctica el alumno podrá:

- Implementar un control de iluminación empleando elementos LDR y el módulo del comparador de voltaje de un microcontrolador usando lenguaje C.
- Construir y calibrar un controlador con punto de ajuste independiente.

Introducción

En la naturaleza, todos los fenómenos que se producen se presentan de manera analógica, es decir, son procesos continuos que van cambiando con el tiempo, por ejemplo, los cambios de temperatura, las ondas de sonido, con su amplitud y frecuencia variables, el flujo del agua, etc.

Los comparadores de voltaje son usados como interfaz entre circuitos analógicos y circuitos digitales, comparan la magnitud de dos voltajes analógicos proveyendo una indicación de tipo digital o binaria en su salida, como se observa en la figura 7.1. La salida de voltaje se modifica dentro de los límites fijados por los voltajes de saturación del amplificador, $+V_{sat}$ y $-V_{sat}$. Aunque se debe de tener en cuenta que la limitante de estos circuitos es que la rapidez de respuesta a su salida es relativamente lenta comparada con otros circuitos.

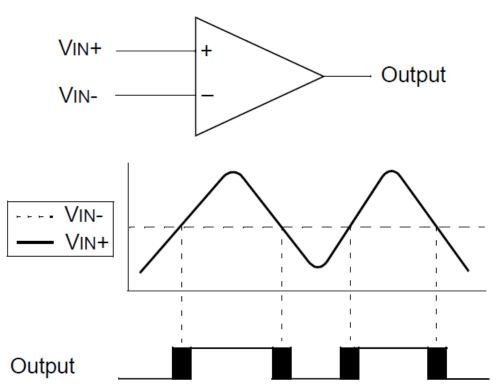


Figura 7.1 Ejemplo de un comparador de voltaje

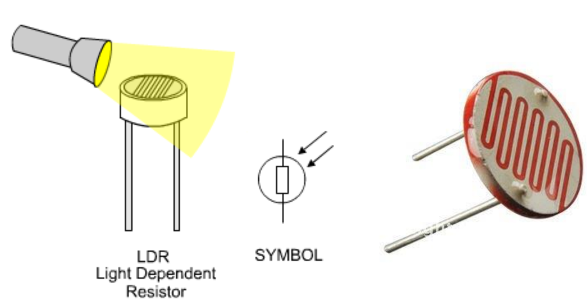


Figura 7.2. Ejemplo de una fotorresistencia LDR.

En el caso de los microcontroladores, los comparadores son muy útiles pues permiten obtener una funcionalidad analógica independientemente de la ejecución del programa.

Los LDR (Light Dependent Resistors), o fotorresistencias, son ampliamente empleados en circuitos de detección de luz/oscuridad. Normalmente la resistencia de un LDR es muy alta, llegando a mega Ohms, pero cuando reciben luz de alguna fuente luminosa, la resistencia cae dramáticamente. Los LDR están fabricados con un material semiconductor de alta resistencia que, al recibir suficiente luz, los fotones son absorbidos por el semiconductor permitiendo que los electrones salten a la banda de conducción del material y creando un efecto de disminución de la resistencia.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

2) Desarrolle un programa que permita que el microcontrolador funcione como un "sistema de control de iluminación". El programa debe hacer uso de los módulos de comparación de voltaje integrados en el microcontrolador, además, el circuito debe contar con entradas de calibración para los voltajes de referencia como se muestra en la figura 7.3. **Revise la hoja técnica del microcontrolador para elegir el modo de operación más apropiado.**

3) El algoritmo debe seguir las siguientes especificaciones:
   - El comparador C1 deberá tener su entrada de voltaje a través del canal 1 (terminal **RA0)** y su referencia deberá estar ligada al exterior (terminal **RA3**) con un voltaje de 2.3V. De acuerdo con el resultado de la comparación, la salida deberá mostrarse por programación en un pin de puerto (señal **C1OUT**) donde deberá estar conectado uno de los LED's. La polaridad seleccionada debe ser no invertida.
   - El comparador C2 deberá tener su entrada de voltaje a través del canal 2 (terminal **RA1)** y su referencia deberá estar ligada al exterior (terminal **RA2**) con un voltaje de 4.2V. De acuerdo con el resultado de la comparación, la salida deberá mostrarse por programación en un pin de puerto (señal **C2OUT**) donde deberá estar conectado a otro LED. La polaridad seleccionada debe ser no invertida.

4) Realice el diagrama de flujo que indica cómo se va a desarrollar el programa.

5) Compile el programa con MPLAB X IDE, corrigiendo los posibles errores.

6) Realice la simulación del sistema de microcontrolador donde se observe en el osciloscopio virtual las tres señales del comparador (entrada variable, referencia fija y salida) para cada comparador, usando tres canales distintos, en el momento del cambio, guarde la simulación y entregue a su profesor una copia impresa o en formato digital de los resultados obtenidos, de acuerdo con lo indicado por el profesor.

7) Incluya junto con las actividades previas, el diagrama de terminales del Led RGB empleado.

8) **El circuito deberá ser armado previamente a la realización de la práctica**.

Equipo

1 PC con software instalado:
- MPLAB X IDE
- Simulador de circuitos electrónicos

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD
1 Multímetro

Material

1 Microcontrolador PIC16F628A
1 Comparardor de voltaje LM339
2 Fotorresistencias LDR de 2 MΩ (también llamada fotoceldas)
3 Potenciómetros de 10 kΩ
2 Resistencias de 1 kΩ a ½ watt
2 Resistencia de 100 kΩ a ½ watt
3 Resistencias de 330 Ω a ½ watt
2 Leds
1 Led RGB de cátodo común
1 Push button
Tableta de Conexiones (Protoboard)
Alambres y cables para conexiones

Procedimiento experimental

1. Haciendo uso del programa de las actividades previas y siguiendo la lógica de su algoritmo programe el microcontrolador.

2. Para el circuito mostrado en la figura 7.3, compruebe que las conexiones correspondan con las terminales de las entradas y salidas de los comparadores de voltaje indicadas en las actividades previas.

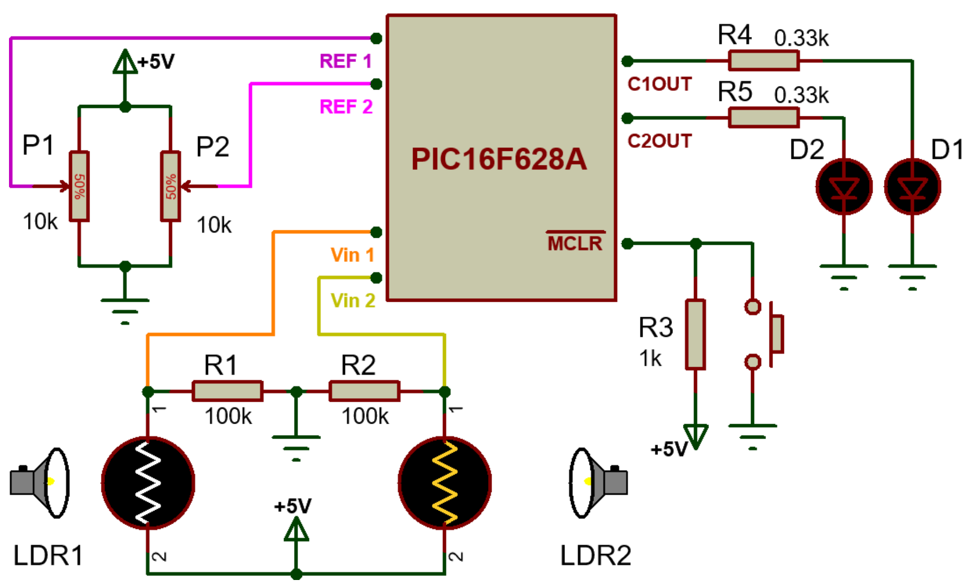


Figura 7.3. Circuito simbólico de control de iluminación.

3. Ajuste los potenciómetros P1 y P2 para igualar los voltajes de referencia Ref 1 y Ref 2 a los indicados en las actividades previas.

4. Acerque una fuente luminosa poco a poco a los sensores de luz y, con ayuda del multímetro, mida el voltaje en el que se produzca un cambio en cada uno de los led's.

5. Repita el punto 4 del desarrollo con la diferencia de que ahora debe comenzar con la fuente de luz sobre los sensores y después alejarla poco a poco. Incluya fotografías del funcionamiento del circuito.

6. **Revise nuevamente la hoja técnica del microcontrolador** para cambiar el modo de funcionamiento de modo que se utilice una sola entrada de referencia para ambos comparadores (referencia común) y que la salida de cada comparador conecte con pines del circuito integrado, es decir, como en el primer programa.

7. Arme el circuito comparador discreto de la figura 7.4, utilizando como entrada Vin3 una señal triangular de 5Vpp y un offset de $2.5V_{CD}$, de modo que no tenga parte negativa, y una frecuencia de 100Hz. La salida del comparador será llamada C3OUT.

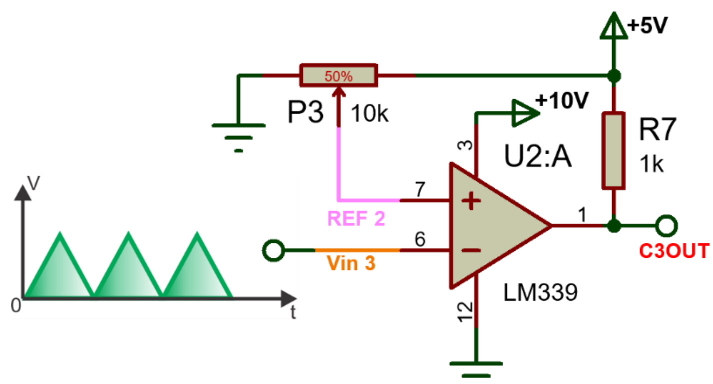


Figura 7.4. Circuito comparador discreto.

8. Desconecte las señales de entrada Vin1 y Vin2 y en su lugar conecte en la terminal común de referencia la misma señal triangular del comparador discreto. Conecte las salidas C1OUT, C2OUT del microcontrolador y C3OUT del comparador discreto a las tres terminales de entrada del LED RGB como se muestra en la figura 7.5.

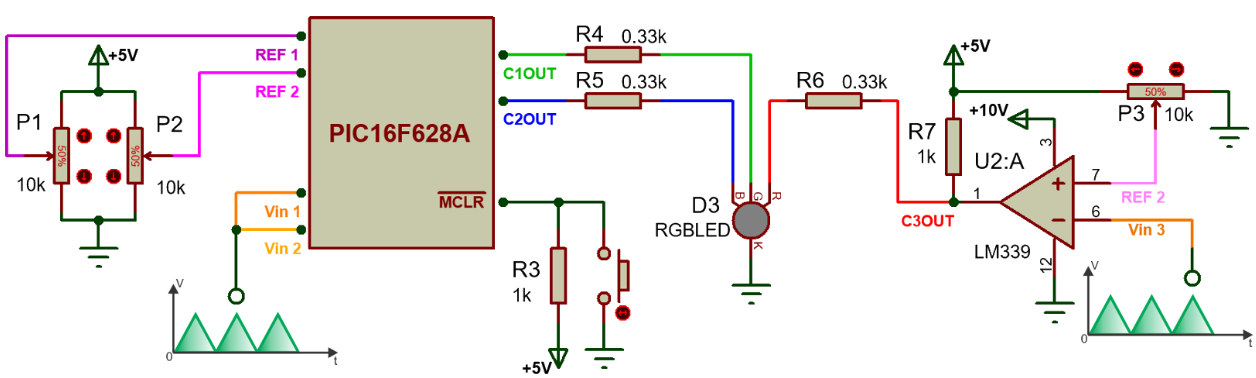


Figura 7.5. Circuito simbólico de control de color.

9. Ajuste los potenciómetros para obtener cada uno de los tres colores independientes que forman al led RGB, es decir, rojo, verde y azul. Incluya imágenes del circuito donde se aprecie el funcionamiento del circuito para estos tres casos.

10. Varíe los voltajes de referencia por medio de los tres potenciómetros para obtener al menos 4 colores diferentes en el led RGB (diferentes a los tres ya obtenidos) y con ayuda del osciloscopio observe y grafique las señales presentes en las terminales de C1OUT, C2OUT y C3OUT, **indicando el ancho de pulso correspondiente**, que corresponden a cada pin del led cuya combinación fue la que proporcionó el color elegido.

11. Anote sus observaciones acerca del comportamiento del led RGB agregando fotografías y explique la razón de dicho comportamiento.

12. Por último, lleve al máximo el ancho de pulso de las tres de salida que van hacia el led para obtener el color blanco, que resulta de la combinación de todos los colores con que cuenta el led e incluya una imagen donde se muestre el resultado.

Cuestionario.

1) Defina el concepto de histéresis en comparadores de voltaje.

2) De acuerdo con las cuatro mediciones realizadas en los puntos 4 y 5 de la práctica, determine el valor de resistencia de los LDR en el momento de los cambios en los Led's e incluya los cálculos realizados.

3) Investigue dos modelos de sensores de luz comerciales e incluya sus principales características como son voltaje y corriente de operación, sensibilidad, etc.

4) Revise los módulos periféricos que contiene el PIC16F887 y explique cuál sería capaz de reproducir las señales observadas en el punto 9 del procedimiento.

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Laboratorio de Microcontroladores
Práctica 8
Convertidor Analógico – Digital

Tema

8.1. Conversión Analógico Digital con microcontroladores.

Objetivos

Al término de esta práctica el alumno podrá:

- Configurar el módulo de conversión analógica – digital con que cuentan los microcontroladores PIC de gama media.
- Implementar un termómetro digital empleando un sensor de temperatura, una pantalla LCD y el módulo del ADC del PIC16F887 usando lenguaje C.

Introducción

En los sistemas digitales, que trabajan únicamente con señales de 1's y 0's, cierto y falso, 0V y 5V, no es posible manejar magnitudes continuas. Para integrar ambos mundos existen dispositivos conocidos como convertidores analógicos – digitales con la capacidad de interpretar señales continuas y traducirlas en valores numéricos binarios que pueden ser comprendidos por los sistemas digitales. El proceso de conversión de una señal se realiza en varias etapas, como se muestra en la figura 8.1. A partir del sistema físico se obtiene una señal con magnitud continua, por ejemplo, la temperatura. Esta magnitud puede ser de naturaleza variada por lo que, para ser compatible con un sistema electrónico, es necesario transformarla a una magnitud eléctrica como son el voltaje o la corriente. Para realizar esta transformación se utiliza un elemento llamado transductor. Un transductor convierte la señal del sensor en una señal de voltaje, que también es analógica.

El siguiente paso es acondicionar la señal, esto normalmente implica etapas de amplificación y filtrado de la señal. Por último, la señal acondicionada entra al convertidor A/D de donde se obtiene una señal digital en formato binario.

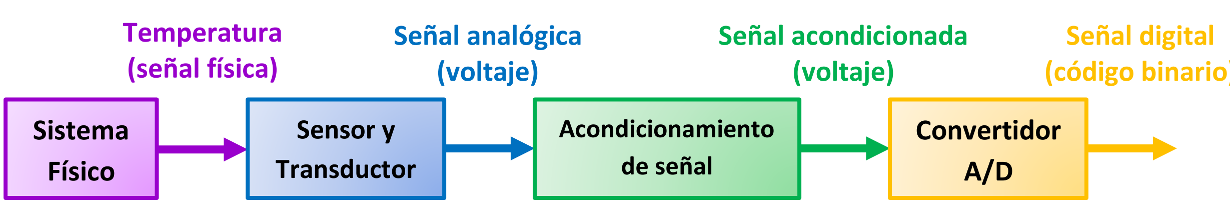


Figura 8.1. Diagrama de bloques del proceso de conversión A/D.

La selección del tipo de convertidor A/D depende de muchos factores que se determinan a través de la naturaleza del sistema físico. Además, se deben tener en cuenta factores como la exactitud, la resolución deseada (el número de bits), el error de cuantización máximo, el tiempo de conversión y la linealidad, que representa la desviación de los códigos de salida respecto al trazo de una línea recta desde cero hasta el valor a plena escala.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

2) Investigue las equivalencias entre las escalas Celsius, Farenheit, Kelvin, Rankine y Reaumur.

3) Desarrolle un programa que, utilizando lenguaje C, permita interpretar señales de voltaje analógico por medio del módulo del convertidor A/D del microcontrolador y mostrar los datos resultantes de la conversión en un display LCD. **Recuerde emplear el driver apropiado para el uso del display LCD**.

4) Dibuje el diagrama de flujo que muestre el desarrollo del programa de conversión analógica – digital.

5) Al desarrollar el algoritmo del programa, todo se reduce a lo siguiente:
   - Declarar las variables que sean necesarias para el desarrollo del programa teniendo cuidado con el tipo de dato que va a asignarse a cada una.
   - Configurar el convertidor A/D del microcontrolador especificando la resolución del convertidor, el canal analógico que se va a utilizar y la fuente de reloj del convertidor.
   - Habilitar el display LCD para comenzar el envío de datos.
   - Crear un ciclo continuo donde se realice el proceso de conversión y se muestre en el display el valor binario que resulta de la conversión además del valor del voltaje analógico de entrada y la temperatura equivalente a ese nivel de voltaje. La figura 8.2 muestra un ejemplo de la información que debe mostrar el display LCD.
   - El sensor empleado puede medir temperaturas de 0°C a 100°C, con un rango de voltaje en su salida de 0V a 1000mV. Debido a que el convertidor se empleará con un voltaje de referencia máximo de 5V entonces se requiere que la señal del sensor se amplifique 5 veces para obtener un rango de 0V a 5V. Los detalles correspondientes al sensor de la práctica se pueden ver en la figura 8.3.
   - No olvide dar un retardo de tiempo entre cada medición para permitirle al usuario leer la información mostrada en el display.
   - Por último, el circuito debe contar con una sola señal de selección para elegir si la temperatura se muestra en Celsius (°C), Fahrenheit (°F) o Kelvin (K) de forma cíclica.

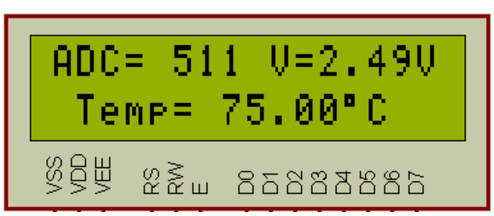


Figura 8.2. Información desplegada en el display LCD después de la conversión A-D.

6) Compile el programa con MPLAB IDE, corrigiendo los posibles errores.

7) Realice la simulación del sistema de microcontrolador basándose en la figura 8.4, guarde la simulación y entregue a su profesor una copia impresa o en formato digital de los resultados obtenidos, de acuerdo con lo indicado por el profesor, donde se aprecien el voltaje a la salida del operacional medido con un multímetro y los datos mostrados en el display para diferentes valores de entrada del sensor de temperatura.

8) **El circuito deberá ser armado previamente a la realización de la práctica**.

$$\boldsymbol{Ganancia[Grados\,{}^\circ\mathrm{C}(Factor\ de\ Escala)] = Voltaje\ del\ Convertidor}$$

$$\mathbf{5}\left[\mathbf{100}\,{}^\circ\mathrm{C}\left(\mathbf{10}\,\frac{\boldsymbol{mV}}{{}^\circ\mathrm{C}}\right)\right] = \mathbf{5}\boldsymbol{V}$$

Figura 8.3. Características del sensor de temperatura LM35D.

Equipo

1 PC con software instalado:
- MPLAB X IDE
- Simulador de circuitos

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD
1 Multímetro

Material

1 Microcontrolador PIC 16F887
1 Sensor de temperatura LM35D
1 Amplificador operacional LM358
1 Display LCD 16x2
1 Resistencias de 10 kΩ a ½ watt
3 Resistencia de 1 kΩ a ½ watt
1 Resistencia de 39 kΩ a ½ watt
1 Resistencia de 82 Ω a ½ watt
1 Potenciómetro de 2 kΩ

1 Capacitor de 1µF a 25V
2 Push button
Tableta de Conexiones (Protoboard)
Alambres y cables para conexiones

Procedimiento experimental

1. Haciendo uso del primer programa de las actividades previas y siguiendo la lógica de su algoritmo programe el microcontrolador.

2. Compruebe que en el circuito de la figura 8.4 la conexión de entrada al microcontrolador que viene desde el sensor sea un pin de puerto con función de entrada analógica y que no interfiera con las terminales que emplea el display, esto para prevenir daños al microcontrolador.

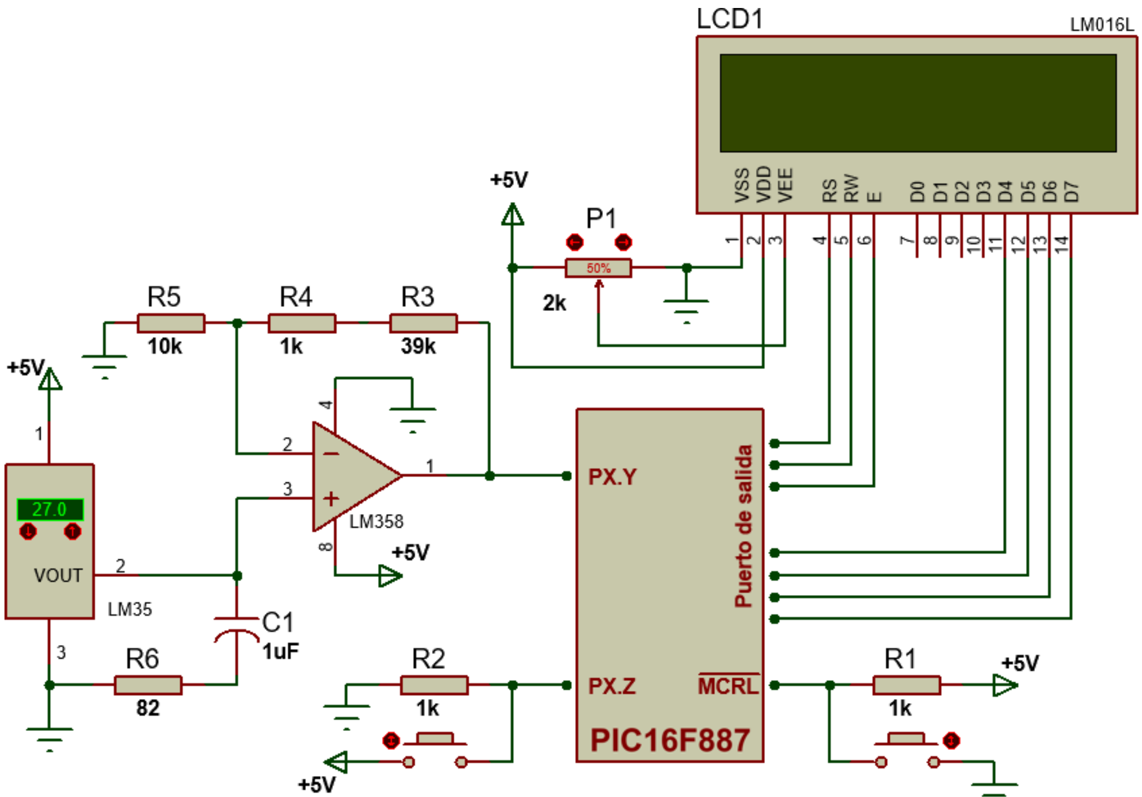


Figura 8.4. Diagrama simbólico del termómetro digital con microcontrolador.

3. Sin tocar el sensor del termómetro digital, anote los datos obtenidos para todas las escalas de temperatura programadas, estos corresponden al valor de la temperatura ambiente.

4. Compruebe con ayuda del multímetro que el voltaje analógico mostrado en el display LCD sea correcto.

5. Modifique el programa para que, empleando el mismo selector, muestre las lecturas de temperatura en grados Rankine y Reaumur además de en grados Celsius del programa original.

6. Utilizando el termómetro digital con el cambio de programación, registre la temperatura de cuatro objetos diferentes, empleando todas las escalas disponibles para cada una, y anótelas en una tabla donde se muestren todos los datos obtenidos.

7. Incluya fotografías del funcionamiento del termómetro digital para las diversas escalas de temperatura.

Cuestionario.

1) Determine el rango de voltaje, la resolución y el error máximo del convertidor A/D empleado, incluya los cálculos realizados.

2) De acuerdo con los resultados de las preguntas anteriores, ¿cuál sería el valor de temperatura mostrado en el display en grados Celsius y el valor de conversión si a la salida del amplificador hay 2.3V? Incluya los cálculos.

3) Explique el funcionamiento de un convertidor A/D de aproximaciones sucesivas.

4) ¿Qué se tiene que modificar en el programa si se desea usar más de un canal analógico?

Conclusiones

Redacte las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Laboratorio de Microcontroladores
Práctica 9
Modulación por ancho de pulso

Tema

9.1. Modulación por ancho de pulso (PWM) empleando microcontroladores.

Objetivos

Al término de esta práctica el alumno podrá:

- Configurar el módulo de Captura, Comparación y PWM (módulo CCP) en modo de generador de señales PWM.

- Implementar un control de velocidad para motores de CD empleando modulación por ancho de pulsos empleando el módulo CCP del PIC16F887 usando lenguaje C.

Introducción

Por lo general, todo sistema que procesa información binaria para controlar un proceso analógico requiere una etapa de entrada analógica – digital y una etapa de salida digital – analógica (convertidores ADC y DAC). Para reducir costos en los diseños que no requieren alta resolución en la etapa de salida, es posible sustituir el DAC por un algoritmo de *Modulación por Ancho de Pulsos* (PWM – Pulse Width Modulation). Una unidad PWM permite asignar cierta duración de tiempo en alto o en bajo a un dato digital de n bits que se considera salida de la etapa de control.

La modulación por ancho de pulsos permite generar señales de frecuencia y ciclo de trabajo variados. Las principales características de una señal de tipo PWM son su periodo y el ciclo de trabajo, como se observa en la figura 9.1.

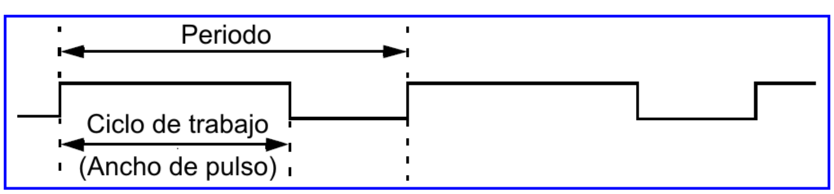


Figura 9.1. Periodo y ciclo de trabajo de la modulación por ancho de pulso (PWM).

El periodo se relaciona con la frecuencia de la señal, lo que indica el número de pulsos generado por unidad de tiempo, mientras que el ciclo de trabajo, también conocido en inglés como *duty cycle*, determina la anchura de cada pulso de manera porcentual respecto al periodo. Hay que remarcar que el ancho del pulso es independiente de la frecuencia de la señal.

El voltaje promedio suministrado por el generador PWM es proporcional a dicha señal, es decir, el voltaje es directamente proporcional ancho del pulso. Como resultado, también la potencia entregada es directamente proporcional, de modo que entre más ancho el pulso (mayor ciclo de trabajo), mayor será la potencia suministrada. Un ejemplo de lo anterior puede observarse en la figura 9.2.

La modulación por ancho de pulso (PWM) es una técnica para controlar circuitos analógicos con una salida digital. Se utiliza en múltiples aplicaciones, algunas como controlar la intensidad de una luz y regular la velocidad de los motores de CD.

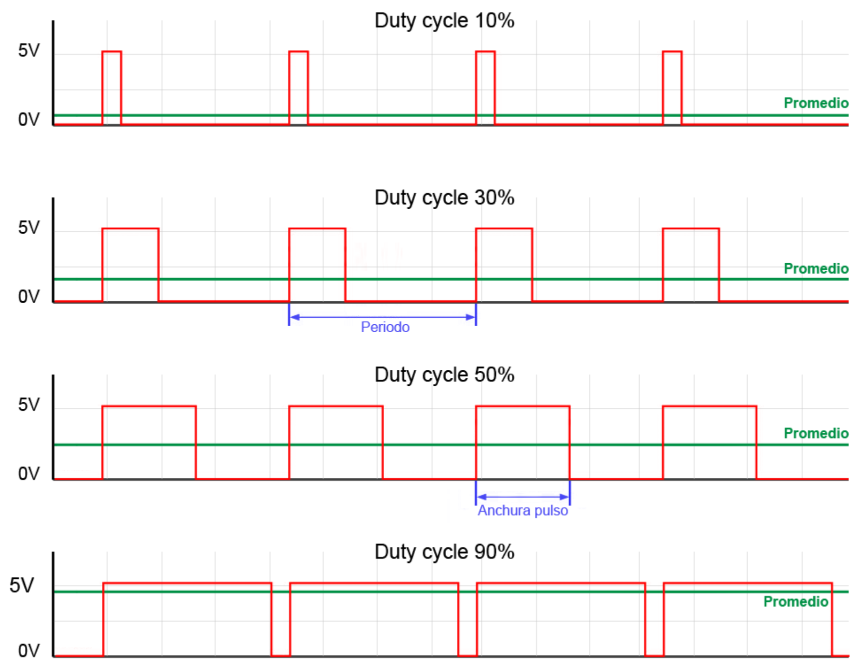


Figura 9.2. Relación Voltaje promedio – Ancho de pulso de una señal PWM.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

2) Realice los cálculos necesarios y escriba un programa para generar una señal PWM mediante el módulo CCP1 del PIC16F628A con las siguientes características: Frecuencia de 6kHz y ciclo de trabajo del 45%. **Recuerde emplear el driver PWM.**

3) Al desarrollar el algoritmo del programa, todo se reduce a lo siguiente:
   - Configurar el puerto C como salida.
   - Configurar el valor del ciclo de trabajo.
   - Configurar el Timer2 para el ciclo de trabajo.
   - Habilitar el módulo CCP1 para que trabaje como PWM.

4) Dentro de las directivas, se tiene que indicar el valor de preescaler a usar, de acuerdo con los cálculos realizados para que el módulo PWM trabaje a la frecuencia deseada:

   #define TMR2PRESCALE X

   donde X puede ser 1, 4 o 16, según el valor de preescaler que se hay utilizado.

5) Para iniciar y detener el módulo PWM, se utilizan las instrucciones:

   PWM1_start(); //inicializa el PWM
   PWM1_stop(); //detiene el PWM

6) Para configurar el periodo de trabajo se utiliza la instrucción:

   PWM1_init(valor de PR2);

   donde el valor de *PR2* se tiene que calcular de acuerdo con la ecuación de ***periodo*** que se indica en las hojas de datos técnicos.

7) Para indicar el ciclo de trabajo se utiliza la instrucción:

   PWM1_Set_duty(valor_ciclo);

   donde *valor_ciclo* se tiene que determinar mediante la ecuación de ***PWM duty cicle*** de la hoja de datos técnicos.

8) Desarrolle un segundo programa cuyo algoritmo proporcione, **de manera automática**, el perfil de velocidad mostrado en la figura 9.3 al presionar el botón de **Inicio**. Se puede apreciar que, del estado de reposo, el motor debe aumentar su velocidad paulatinamente en dos fases hasta el punto deseado, mantener esa velocidad durante un periodo de tiempo y luego desacelerar hasta terminar en la posición de reposo.

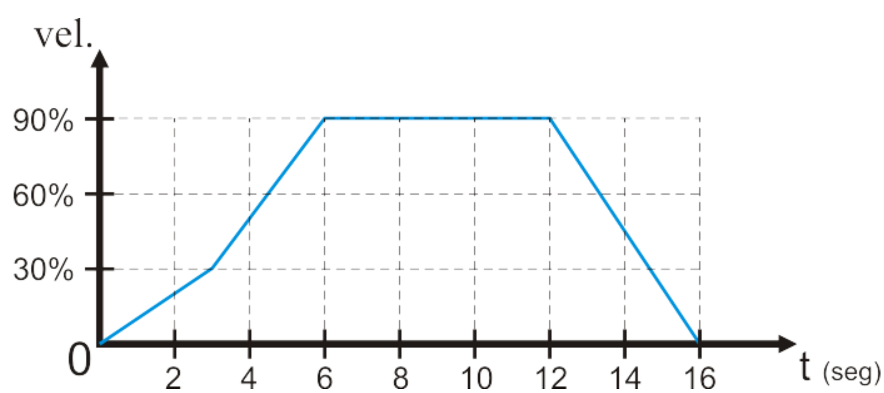


Figura 9.3. Perfil de velocidad controlado mediante PWM.

9) El programa debe iniciar al presionar un botón de arranque y debe de permitir seleccionar el sentido de giro del motor por medio de un selector que controle las señales D1 y D2.

10) Realice la simulación del sistema de microcontrolador basándose en la figura 9.5, guarde la simulación y entregue a su profesor una copia impresa o en formato digital de los resultados

obtenidos, de acuerdo con lo indicado por el profesor, donde se aprecien el voltaje a la salida del módulo PWM en diferentes anchos de pulso.

11) **El circuito deberá ser armado previamente a la realización de la práctica**.

Equipo

1 PC con software instalado:
- MPLAB X IDE
- Simulador de circuitos

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD
1 Osciloscopio
1 Multímetro

Material

1 Microcontrolador PIC 16F628A
2 Resistencia de 10 kΩ a ½ watt
4 Diodos 1N4004
1 Driver L298
1 Push button
1 Motor de CD (de 10V o 12V)
Tableta de Conexiones (Protoboard)
Alambres y cables para conexiones

Nota: Puede sustituirse el controlador L298N y los diodos 1N4004 por un módulo integrado como el que se mostró en la figura 6.4 de la práctica 6.

Procedimiento experimental

1. Haciendo uso del primer programa de las actividades previas y siguiendo la lógica de su algoritmo programe el microcontrolador.

2. Proceda a armar el circuito en la protoboard, basándose en el mostrado en la figura 9.4, tenga cuidado de que la salida corresponda con la terminal de salida del módulo CCP1.

3. Encienda el circuito y con ayuda del osciloscopio, mida la terminal de salida para comprobar que la señal tiene la frecuencia y el ciclo de trabajo indicados en las actividades previas y grafíquela.

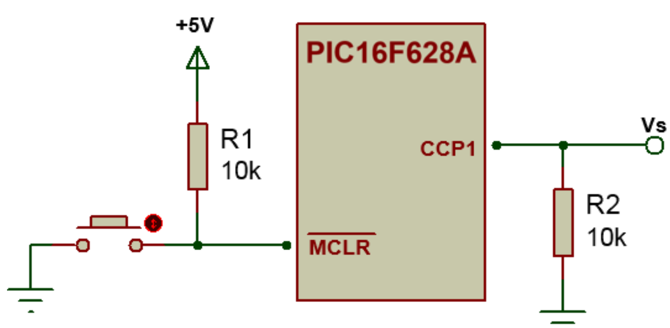


Figura 9.4. Diagrama simbólico del generador de señales PWM.

4. Programe nuevamente el microcontrolador, ahora con el programa del perfil de velocidades.

5. Modifique el armado del circuito según se muestra en la figura 9.5.

6. Encienda el sistema y observe si el comportamiento del motor corresponde con el esperado, **anote sus observaciones**.

7. Con ayuda de un multímetro mida la corriente y voltaje promedio suministrados al motor en cada fase del perfil de velocidades. Anote cada uno de los valores.

8. Incluya fotografías del funcionamiento del sistema de regulación de velocidad.

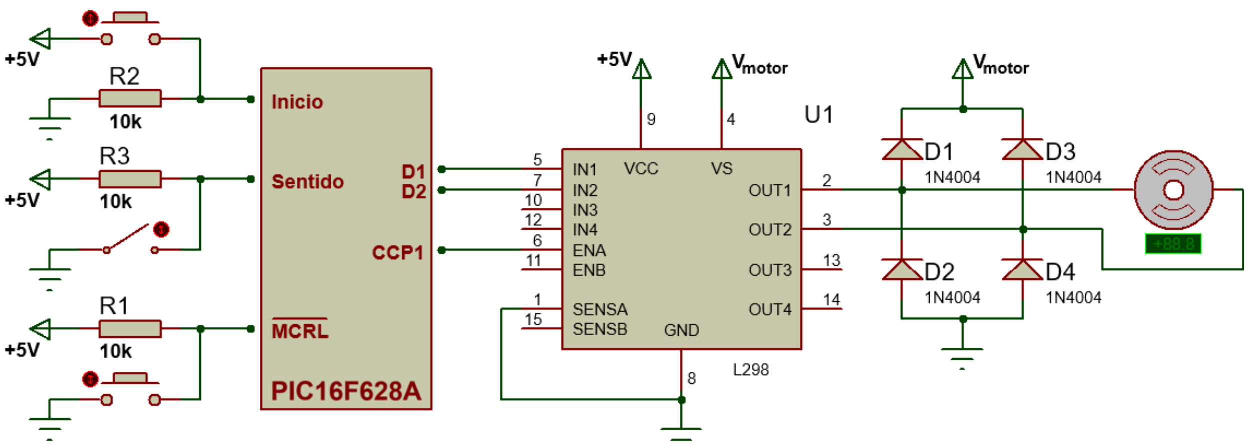


Figura 9.5. Circuito simbólico de control de velocidad para motor de CD.

Cuestionario.

1) Para el primer programa de las actividades previas, si se deseara cambiar el ciclo de trabajo entre el valor original y el 45% por medio de un selector, explique qué modificaciones tendría que hacer usted al código y por qué e incluya el programa ya modificado.

2) Investigue y explique la forma en que se puede controlar la posición angular de un servomotor empleando señales PWM.

3) Empleando los datos obtenidos en el punto 7 del desarrollo, calcule la potencia media suministrada al motor para cada ancho de pulso utilizado.

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.

Laboratorio de Microcontroladores

Práctica 10

Comunicación serial – Protocolo I2C

Tema

9.6. Otros recursos.

Objetivos

Al término de esta práctica el alumno podrá:

- Comprender el uso del módulo de comunicación serial del microcontrolador.
- Hacer uso del protocolo y puerto de comunicaciones I2C en conjunto con una memoria EEPROM de tipo serial.

Introducción

La comunicación serial es un medio de intercambio de información entre dos dispositivos siguiendo estándares establecidos, como pueden ser el RS-232, de uso comercial y el RS-485 de uso industrial. La comunicación serial es un método simple que existe desde hace varios años y que se mantiene vigente gracias a su simplicidad en el manejo de la información para su envío y recepción.

La comunicación serial puede dividirse en dos tipos, la comunicación **asíncrona**, también llamada *full-duplex* y la comunicación **síncrona**, conocida como *half-duplex*.

La forma más habitual de la comunicación serial es la de tipo asíncrona la cual esta representada de manera simbólica en la figura 10.1, donde se muestra la conexión entre dos dispositivos en los que ***TX*** representa la terminal de transmisión y ***RX*** la terminal de recepción de cada uno, además de que comparten la terminal de tierra (GND).

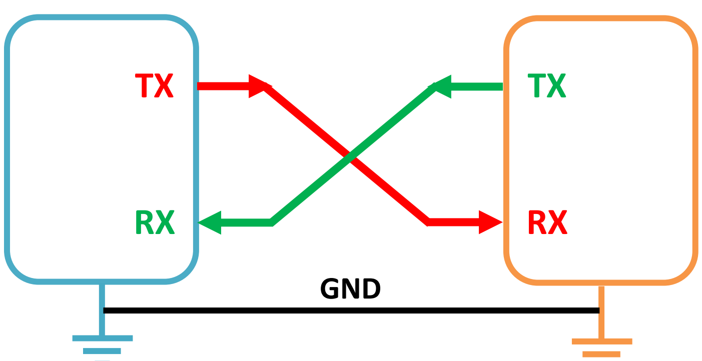


Figura 10.1. Comunicación serial asíncrona (full-duplex).

En el caso de la comunicación síncrona (half-duplex) se requiere de un canal adicional que sirve como medio de sincronización entre los dos dispositivos enviando señales de reloj digital. Adicionalmente, se debe de definir cuál de los dos dispositivos actuará como el **maestro** (*master*) y cuál será es **esclavo** (*slave*), esto simplemente significa que el maestro será el que emita la señal de reloj y el esclavo se limitará a seguir la señal de sincronización, como se observa en la figura 10.2.

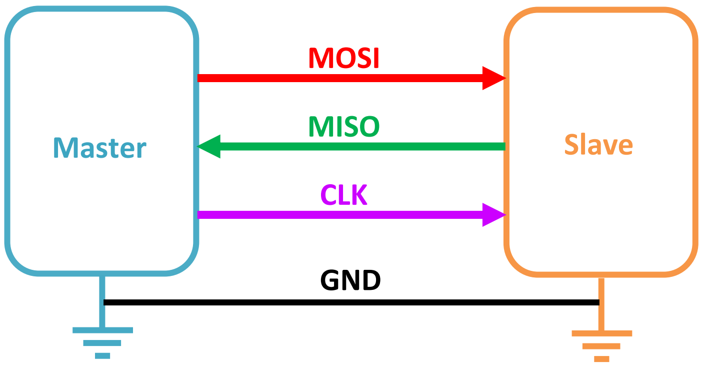


Figura 10.2. Comunicación serial síncrona (half-duplex).

Las líneas MOSI y MISO son los acrónimos de *Master Output – Slave Input* y de *Master Input – Slave Output* respectivamente, lo cual se puede traducir como Salida del Maestro – Entrada del Esclavo y Entrada del Maestro – Salida del Esclavo, una descripción de su funcionalidad.

Algunos microcontroladores disponen del módulo de comunicación serie USART/SCI, tal vez el más utilizado entre los módulos de interfaz serie y que se puede traducir como *Módulo del Receptor Transmisor Síncrono Asíncrono Universal* (USART por sus siglas en inglés) o simplemente módulo de comunicación serial. USART es también conocido como interfaz de Comunicación Serial o SCI por sus siglas en inglés.

El USART puede ser configurado como un sistema asíncrono que puede comunicarse con dispositivos como las computadoras personales, o puede ser configurado como un sistema síncrono que puede comunicarse con dispositivos periféricos, como circuitos integrados A/D o D/A, memorias EEPROM seriales, etc.

Existe otro módulo periférico de comunicación serial que pueden contener los microcontroladores dependiendo de sus características llamado MSSP (por sus siglas en inglés *Master Synchronous Serial Port)*, Puerto Serial Síncrono Maestro. Este módulo periférico puede trabajar con memorias EEPROM seriales, registros de corrimiento, controladores de displays, convertidores A/D, etc.

El módulo SSP puede operar en dos modos distintos:

- Interfaz Serie de Periféricos (SPI), desarrollada por la empresa Motorola para comunicación entre microcontroladores aún siendo de distintas familias en modo full-duplex.
- Interfaz Inter Circuitos (I2C o I^2C), desarrollada por Philips con la capacidad de comunicar microcontroladores y periféricos en modo half-duplex.

La comunicación I2C, al ser half-duplex, requiere de dos líneas de comunicación de colector abierto, por lo que requieren el uso de resistencias de elevación (*pull-up*), que son la señal de reloj SCL (*Serial CLock*) y SDA (*Serial Data*). Cada dispositivo conectado al bus I2C tiene asignada una dirección de memoria, dependiendo de la dirección que se le asigne tendrá la función de maestro o esclavo. Hay que recordar que, el dispositivo maestro es el comienza la comunicación y decide el orden en que se realiza la transferencia de información, la dirección de esta y cuándo finaliza. Cuando el maestro inicia la comunicación envía la dirección del dispositivo con el cual se va a comunicar y el esclavo revisa si es su dirección o pertenece a otro dispositivo. La transmisión puede ser de lectura o escritura y esto se indica en el último bit que compone el valor numérico de la dirección.

Actividades previas a la práctica

1) Realice la lectura de la práctica.

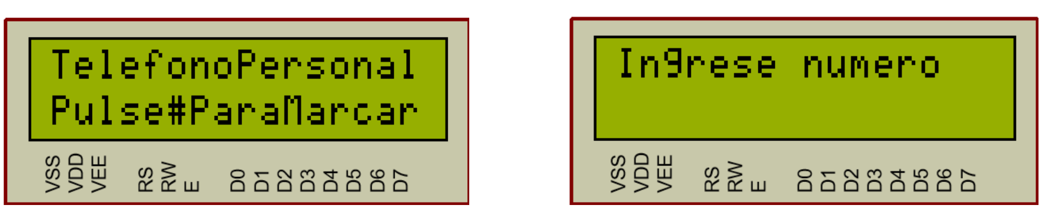
2) Usando el microcontrolador indicado en la lista de material de la práctica, desarrolle un programa en **lenguaje C** que permita simular un sistema de marcado telefónico de 10 dígitos empleando una memoria de tipo EEPROM serial que servirá para almacenar el número que se desea marcar.

3) El desarrollo del algoritmo se reduce a:
   - Configurar el módulo de comunicación I2C
   - Desarrollar un programa que contenga tres funciones principales, una función de escritura en la memoria, una función de lectura de la memoria y la función *main*.
   - La primera función que debe aparecer es la función de escritura de la memoria EEPROM, esta función tiene un conjunto de instrucciones específicas que se pueden encontrar en las fuentes bibliográficas.
   - La segunda función que debe aparecer es la de la lectura de la memoria EEPROM, de manera similar a la escritura, tiene una forma específica.
   - Finalmente, la función *main* se encargará de administrar el teclado, el display LCD y la entrada y salida de datos además de permitir corregir el número telefónico que se está ingresando. Esta última es la que requiere mayor atención pues no hay una forma única de crear la función.

4) Recuerde que las terminales en el bus de comunicación I2C son de colector abierto, por lo que requieren el uso de resistencias de elevación (pull-up).

5) Al inicio, el sistema debe solicitar el ingreso del número telefónico de 10 dígitos por medio de la tecla #, como se presenta en la figura 10.3a.

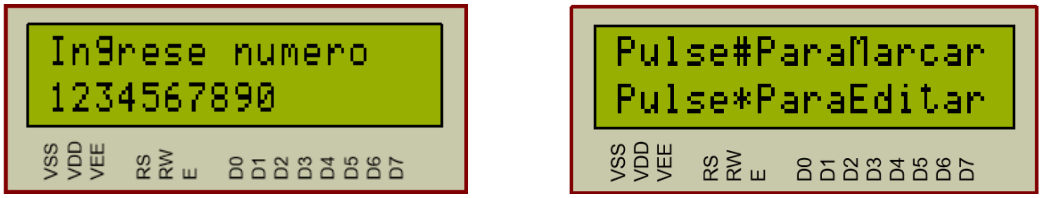
6) Una vez que se presione la tecla #, el microcontrolador debe prepararse para recibir el número telefónico e ingresarlo en la memoria EEPROM, ver figura 10.3b.



(a) (b)

Figura 10.3. Inicio del sistema.

7) Cada que se presione una tecla el número debe mostrarse en el display LCD, como se observa en la figura 10.4a, y cuando se completen los diez dígitos del número, el sistema deberá preguntar si se realiza el marcado o hay un error y se desea reingresar el número, figura 10.4b.



(a) (b)

Figura 10.4. Ingreso del número y corrección.

8) Por último, si se eligió editar el número el sistema regresa a la pantalla de "Ingrese número" y vuelve a preguntar si es correcto para marcar. Si se confirma la marcación entonces el sistema deberá leer desde la memoria EEPROM el número como si estuviera haciendo la marcación y mostrarlo en el display LCD, como se muestra en la figura 10.5.

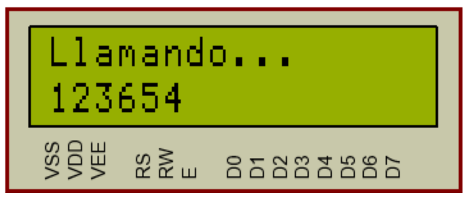


Figura 10.4. Lectura de la memoria EEPROM.

9) **El circuito deberá ser armado previamente a la realización de la práctica**.

Equipo

1 PC con software instalado:
- MPLAB X IDE
- Simulador de circuitos

1 Grabador universal o grabador de PICs
1 Fuente de voltaje de CD
1 Osciloscopio
1 Teclado matricial

Material

1 Microcontrolador PIC16F887
1 Memoria 24LC512
5 Resistencias de 10 kΩ, ½ watt
2 Resistencias de 4.7 kΩ, ½ watt
1Potenciómetro de 2kΩ
1 Push button
1 Display LCD 16x2
Tableta de conexiones
Alambres y cables para conexiones

Procedimiento experimental

1. Arme el circuito mostrado en la figura 10.6 teniendo cuidado al manipular la memoria EEPROM 24LC512 para evitar dañarla.

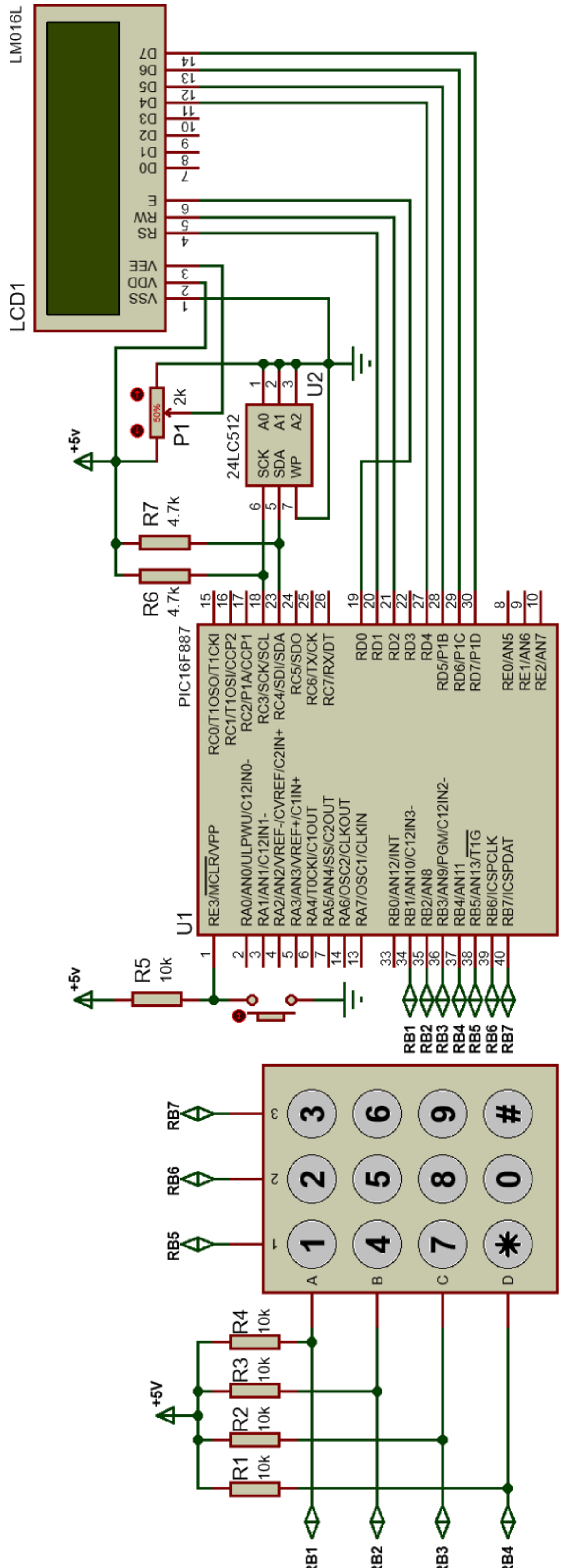


Figura 10.6. Comunicación I2C con la memoria EEPROM.

2. Haciendo uso del proyecto creado anteriormente y siguiendo la lógica del algoritmo de las actividades previas, programe el microcontrolador.

3. Verifique que el circuito realice la secuencia de forma correcta como se describió en las actividades previas, confirmando que se desea ingresar el número y luego confirmando que se realice la marcación. Tome fotografías del funcionamiento.

4. Pruebe que también funcione la opción de editar el número antes de marcar. Tome fotografías del funcionamiento.

Cuestionario.

1) ¿Con qué otros dispositivos es posible establecer comunicación usando el módulo periférico I2C? De al menos dos ejemplos.

2) Si se usara un esquema de conexión I2C *multislave*, ¿cuántos dispositivos se pueden llegar a conectar al módulo del microcontrolador?

3) Explique cómo es que se asignan las direcciones de memoria para determinar si el dispositivo con el que se establece la comunicación es master o slave.

Conclusiones

Elabore un resumen que muestre las conclusiones a las que haya llegado después de realizar todas las actividades de esta práctica.

Bibliografía

Elabore una lista de las referencias bibliográficas consultadas.