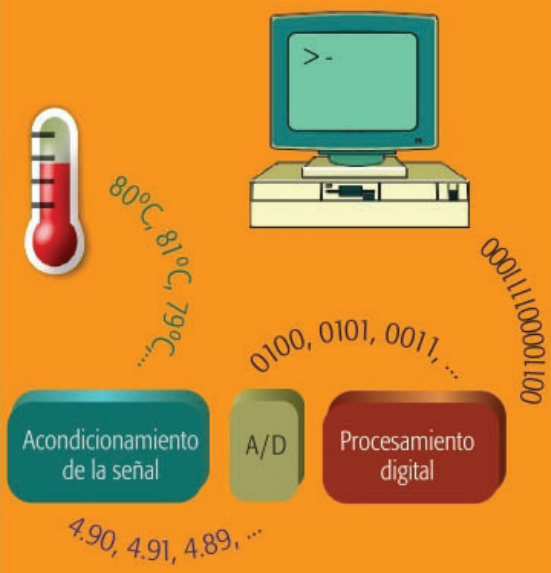
colección **notas de clase**

ELECTRÓNICA DIGITAL Y SU APLICACIÓN A LA INSTRUMENTACIÓN

[guía de laboratorio]

Javier Fernando Cardona
Óscar Roberto Blanco



UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE BOGOTÁ
FACULTAD DE CIENCIAS

Guía de Laboratorio para el Curso de Electrónica Digital

Departamento de Física
Facultad de Ciencias
Universidad Nacional de Colombia

Ph.D. Javier Cardona
Ing. Oscar Blanco

6 de septiembre de 2011

Índice general

Índice de figuras

Índice de tablas

Prólogo

La instrumentación moderna tanto para los físicos como para otros científicos ha crecido enormemente en versatilidad, complejidad y confiabilidad en gran parte debido a la introducción de sistemas digitales. Consideramos entonces que aquellos profesionales de la ciencia cuyo trabajo involucre medición y procesamiento de datos deberían tener un conocimiento mínimo de electrónica y en particular de electrónica digital.

En un sistema de medición moderno pueden identificarse partes como el sensor, la electrónica de acondicionamiento del sensor, el conversor análogo digital, el sistema de procesamiento de datos digitales y visualización, el sistema de adquisición de datos y finalmente el sistema de cómputo. Excepto por el sensor y la electrónica de acondicionamiento todas las partes mencionadas hacen parte del dominio de la electrónica digital.

En estas guías de laboratorio se presentan prácticas básicas que ayudan al estudiante a comprender los principios de la electrónica digital combinacional y digital secuencial. Estas prácticas están diseñadas de tal forma que se van introduciendo simultáneamente las partes básicas del sistema digital que mas conocemos y empleamos en nuestra vida diaria: el computador.

El curso regular de electrónica digital también incluye el conocimiento y manejo de microcontroladores por lo que estas guías incluyen varias prácticas que han sido diseñadas con el microcontrolador 16f877. Los microcontroladores no son más que computadores con todas las partes incluidas en un solo circuito integrado pero con menor capacidad que los computadores regulares que conocemos. Muchos de estos microcontroladores o PICs tienen módulos de electrónica adicionales como temporizadores, conversores análogo digital, etc. que los convierte en dispositivos muy funcionales. Por ejemplo, toda la electrónica digital de un sistema de medición desde el conversor analógo digital hasta el computador puede reemplazarse por uno de estos microcontroladores debidamente programado.

Para asegurar que el curso sea autocontenido se han añadido 2 prácticas de electrónica analógica que pretenden darle la formación mínima al estudiante para que pueda diseñar los circuitos que convierten la señal análoga del sensor ó transductor eléctrico en el rango de voltajes que puede manejar un conversor análogo-digital, usualmente de 0 a 5 voltios.

Las prácticas mencionadas hasta el momento constituyen el núcleo básico de este curso y están organizadas en la primera parte de este libro bajo el nombre de "Prácticas Básicas". Las "Prácticas Básicas" dan al estudiante las herramientas y habilidades mínimas para entender y construir un sistema de medición moderno como el descrito en un párrafo anterior.

Las "Prácticas Adicionales" permiten profundizar en otros tópicos tales como la visualización de datos en un display digital, manejo de interrupciones y manejo de los puertos paralelo y serial de un computador para transmisión de datos.

En los "Apéndices" se puede encontrar información acerca de aspectos del curso como la forma sugerida de presentar los informes, el proyecto final, descripción de los equipos del laboratorio y recomendaciones para su uso, así como también se describe la forma como deben instalarse los programas que se utilizan para programar los "pics".

Cada una de las prácticas sigue un formato estándar que consta de al menos 5 secciones tales como los objetivos de la práctica, materiales necesarios para realizar la práctica, lo que se debe saber antes de poder hacer la práctica, una breve introducción al tema de la práctica y el procedimiento a seguir para el desarrollo de la práctica. La introducción no es bajo ninguna circunstancia un reemplazo de la clase teórica correspondiente ni tampoco lo es este texto. En la sección "¿Qué debe saber antes del laboratorio?" se hace una descripción detallada de la bibliografía que el estudiante debería leer y/o la clase a la que debió haber asistido antes de poder realizar la práctica.

Estas guías de laboratorio han sido utilizadas por varios semestres en el curso de electrónica digital de la Carrera de Física junto con clases teóricas tal como se muestra en la tabla 1. A pesar del corto tiempo disponible (16 semanas) los estudiantes no solo reciben una visión completa de un sistema de medición, sino que además adquieren habilidades para construir y modificar sus propios sistemas y así solucionar problemas típicos que se presentan en nuestros laboratorios de investigación a un muy bajo costo. Ejemplos de algunos proyectos de fin de curso que han logrado solucionar problemas reales y los hemos mostrado en congresos nacionales se describen en el apéndice D.

Algunas de la prácticas mostradas en este texto son adaptaciones de

Clase de Electrónica Analógica	Semana 1
Laboratorio de Transistores	Semana 2
Laboratorio de Operacionales	Semana 3
Clase de Lógica Combinacional Discreta	Semana 4
Clase de Lógica Combinacional MSI	Semana 5
Laboratorio de ALU	Semana 6
Clase de Lógica Secuencial	Semana 7
Laboratorio de Lógica Secuencial	Semana 8
Clase de Microcontroladores	Semana 9
Laboratorio de Microcontroladores	Semana 10
Laboratorio de Temporizadores	Semana 11
Clase de Conversión A/D y RS232	Semana 12
Laboratorio de Conversión A/D y RS232	Semana 13
Proyecto Final	Semanas 14 a 16

Tabla 1: Cronograma recomendado

prácticas que se hacían antes del primer semestre del 2006 cuando los profesores Luis Gutierrez y Ramiro Cardona estaban a cargo del curso de electrónica digital para la Carrera de Física. En particular, la práctica 4 y la práctica 5 han sido adaptadas de prácticas que el profesor Luis Gutierrez solía hacer. Por otro lado, el profesor Ramiro Cardona nos indicó como usar el display de la práctica 10 y nos cedió amablemente el código que escribió para manejar este display. A ellos, nuestros más sinceros agradecimientos por el apoyo brindado en todo lo relacionado con este curso.

Parte I

Prácticas Básicas

Práctica 1

Transistores

1.1. Objetivos

- Introducir al estudiante a los conceptos de amplificación lineal, corte y saturación en un transistor bipolar.

1.2. Materiales

- 1 Oscilocopio
- 1 Generador de Ondas
- 1 Voltímetro
- 2 Sondas
- 1 Fuente de 5 voltios
- 1 Protoboard
- 2 Transistores 2N2222
- resistencias de varias denominaciones

1.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- El transistor sin polarización.
- El transistor polarizado.
- Corrientes en un transistor.
- La conexión con emisor común.
- Curva característica de la base.
- Curva característica del colector.
- Aproximaciones de los transistores.
- Interpretar las hojas de datos.
- Variaciones de la ganancia de corriente.
- La recta de carga.
- El punto de trabajo.
- Reconocer la saturación.
- El transistor en conmutación.
- Polarización de emisor.

Estos temas corresponden a la primera clase de teoría y/o haber leido los secciones 6.1 a 6.8 y 7.1 a 7.6 del libro de Malvino (quinta edición)[1].

Antes de iniciar este laboratorio lea cuidadosemente el Anexo B, allí encontrara instrucciones específicas de como manejar los diferentes instrumentos que se utilizarán en este laboratorio.

Leer la hoja de datos del transistor 2N2222([2]).

1.4. Introducción

El transistor es un dispositivo electrónico que controla la corriente que fluye entre dos de sus terminales, de acuerdo a un valor de voltaje[1] (Transistor

[1]En realidad el aumento del voltaje se debe a la inyección de corriente en una juntura que tiene el comportamiento de un condensador, sin embargo, es válido decir que se activa al llegar a un nivel del voltaje determinado para cada componente.

de efecto de campo, FET) o corriente (Transistor de unión bipolar, BJT) en su tercera terminal.

Para el transistor de unión bipolar, las terminales son conocidas como:

- Emisor: Emisor de cargas en el flujo electrónico.
- Colector: Colector de cargas en el flujo electrónico.
- Base: Terminal de control.

Para un transistor BJT, la relación entre la corriente por la terminal de control I_b y la corriente por la terminal colectora I_c está dada por la ecuación 1.1.

$$I_c = \beta I_b \tag{1.1}$$

donde β es un parámetro que toma valores cercanos a 100 o mayores y es característico de cada componente.

El transistor puede ser utilizado como amplificador, oscilador, conmutador, rectificador,[2] u otros, dependiendo del método y la zona de polarización en la cual se encuentre operando.

La operación del transistor divide el diagrama de polación en tres zonas:

- Lineal: Es la zona sobre la cual la razón de cambio entre la corriente en la base y la corriente en el colector se relacionan linealmente, de acuerdo a lo expresado en la ecuación 1.1. Esta zona es utilizada para la relización de amplificación de señales.
- Saturación: Es la zona sobre la cual la corriente a través del colector ha llegado al máximo posible por el dispositivo, no tiene más carga que aportar a la corriente y por lo tanto, los cambios en la corriente de base no aumentan la corriente de colector. Normalmente utilizado junto con la zona de saturación para funcionar como interruptores, en este caso un interruptor cerrado (máxima corriente).
- Corte: Es la zona sobre la cual la corriente a través del colector llega a un mínimo (incluso cero), debido a que la polarización de la juntura no es suficiente para permitir el paso de carga eléctrica, por lo cual, cualquier cambio de corriente en la base no presenta cambio en la corriente de colector. Es por esta razón que se utiliza como interruptor abierto (corriente muy baja o nula).

[2] Tomado de [3].

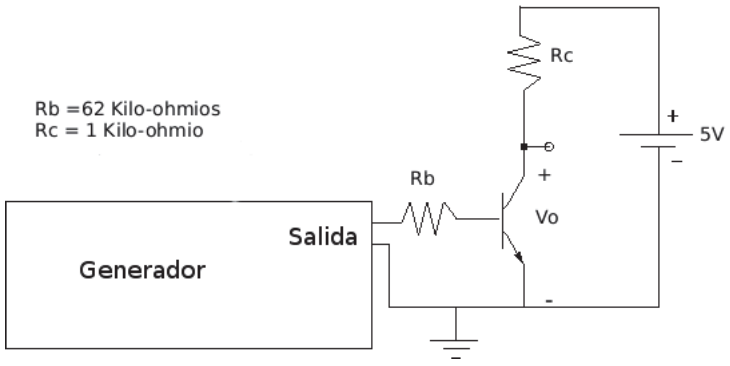


Figura 1.1: Polarización de un transistor.

Para cada una de las funciones anteriores existen diferentes formas de configuración que logran diferentes características en el circuito. En este caso nos referiremos únicamente a la configuración en emisor común.

Se denomina emisor común (Figura 1.1) debido a que es punto común entre la entrada (la base) y la salida (el colector). Realizando un modelo del transistor para pequeña señal y tomando en cuenta solo frecuencias medias (entre 100 Hz a 100 KHz), se puede hallar una relación simple de ganancia de voltaje[3] entre la entrada de señal y la salida. El valor de esta ganancia está determinado por:

- La ganancia del transistor: β.
- El circuito de polarización.
- La resistencia de entrada y salida equivalentes del transistor.

1.5. Procedimiento

1.5.1. Regiones de Operación del Transistor

- Arme en el protoboard el circuito que se muestra en la Figura 1.1. Tenga en cuenta que para el 2N2222[4] el pin del extremo izquierdo (vista con el lado plano del transistor hacia el frente) es el emisor, la pata del medio es la base y la pata del extremo derecho es es el colector. Asegurese que el generador esté encendido y que ningún botón de funciones este oprimido. Hale la perilla DC offset y mida el voltaje de salida del generador. Inicialmente ajuste la perilla de tal forma que el voltaje DC sea de 0 voltios, incremente lentamente el voltaje mientrás mide el voltaje a través de la resistencia del colector. Si todo funciona correctamente el voltaje en Rc cambiará casi linealmente a medida que se aumenta el voltaje del generador hasta que el voltaje a través de Rc se haga igual al voltaje de la fuente de alimentación. Después de este punto, el voltaje a través de Rc se mantendrá casi constante. Haga una tabla del voltaje del generador, el voltaje a través de Rc y el voltaje a través de Rb. A partir de esta tabla haga una gráfica de Ic vs Ib. Identifique la región en la cual Ic es proporcional a Ib (región activa) y a partir de allí calcule el valor beta del transistor. Determine la corriente máxima del colector que se alcanza y compare con el valor teórico calculado en clase. A dicha corriente se le conoce como la corriente de saturación y a la región operación donde Ic es igual a la corriente de saturación se le conoce como la región de saturación. La región de corte corresponde a los primeros puntos de la gráfica donde Ic es cero hasta que se alcanza cierto valor de voltaje en el generador.

1.5.2. El Transistor como Amplificador de Señales

- Ajuste el voltaje DC del generador de tal forma que el voltaje en Rc sea igual a la mitad del voltaje de la fuente (2.5 V). Mida la corriente de base y de colector a través de mediciones de votaje hechas en Rc

[3]Entiendase ganancia como la razón entre el valor a la salida y el valor en la entrada, para el caso de ganancia de voltaje y ganancia 10 significa que el voltaje de salida es igual a diez veces el voltaje de la entrada.

[4]Hoja de datos [2]

y Rb y mida también el voltaje del generador. Usando como variables conocidas el voltaje del generador, el valor de las resistencias, y el beta del transistor calcule la corriente de base, la corriente de colector y el voltaje colector-emisor. Compare con los resultados obtenidos por medición. A este conjunto de valores se les conoce como el punto de trabajo o punto Q del transistor. Conservando el punto Q hallado en el paso anterior, encienda en el generador una señal sinusoidal de 100 mV pico a 1 Khz. Usando el osciloscopio mida la señal del generador y comparela con la señal a la salida del circuito (colector-tierra). ¿Cuántas veces se amplificó la señal?. ¿Corresponde esta amplificación con el cálculo teórico?. Aumente la amplitud de la señal del generador hasta que la onda a la salida del circuito empiece a recortarse. ¿Qué sucede?. Explique. Usando nuevamente la señal sinusoidal de 100 mV pico a 1Khz, aumente el voltaje DC del generador hasta que la señal se recorte y explique. Haga lo mismo disminuyendo el voltaje DC del genderdor.

1.5.3. EL transistor como Compuerta Lógica

- Conecte el punto A del circuito de la figura 1.2 al terminal positivo de la fuente de 5 voltios. Bajo estas condiciones el transistor debería encontrarse en su región de saturación. Compruebelo experimentalmente (midiendo voltajes en los lugares apropiados), haga los cálculos teóricos (suponiendo saturación ideal) y explique las diferencias con las mediciones experimentales. Ahora conecte el punto A al terminal negativo de la fuente de 5 voltios. Bajo estas condiciones el transistor debería estar en su región de corte. Compruebelo experimentalmente y teóricamente. Este sencillo circuito es la base de la electrónica digital y recibe el nombre de compuerta negadora o compuerta NOT. ¿Por qué?

  Ayuda: La electrónica digital a diferencia de la electrónica analógica admite solo 2 valores de voltaje, 0 voltios y 5 voltios a los cuales se les denomina 0 lógico y 1 lógico respectivamente. Usando esta información haga una tabla de las posibles entradas del circuito de la figura 1.2 y sus correspondientes salidas (tabla de verdad).

- Arme el circuito de la figura 1.3 y elabore su tabla de verdad (note que las entradas A y B dan lugar a 4 valores posibles de entrada). A que compuerta lógica corresponde?.

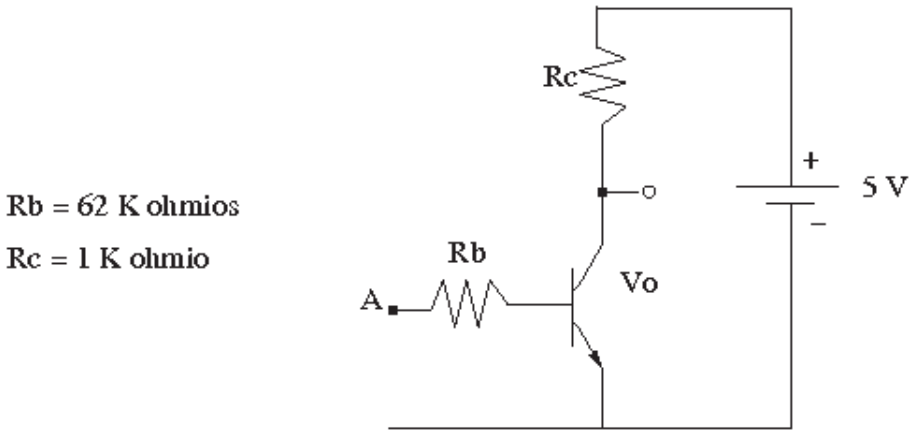


Figura 1.2: El transistor como compuerta lógica.

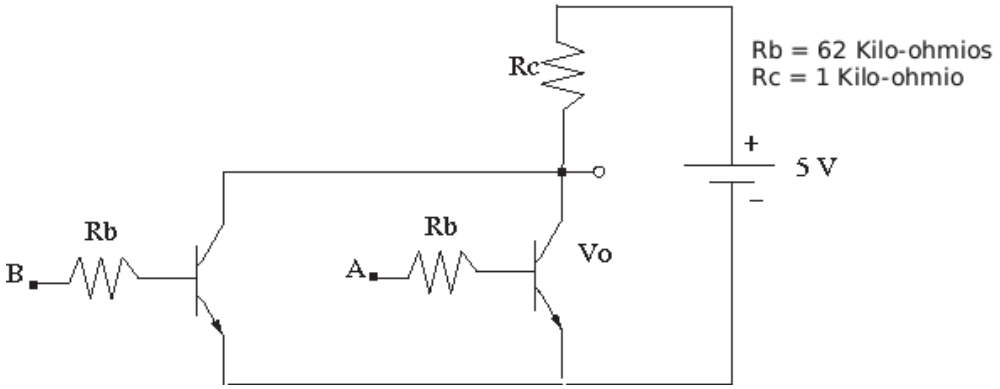


Figura 1.3: ¿Qué compuerta lógica es?

Práctica 2

Amplificadores Operacionales (OPAM)

2.1. Objetivos

- Introducir el uso del amplificador operacional como un circuito amplificador de señales y como un circuito comparador.

2.2. Materiales

- 1 Oscilocopio
- 1 Generador de Ondas
- 1 Voltímetro
- 2 Sondas
- 1 Fuente de 5 voltios
- 1 Protoboard
- 1 Amplificador operacional LM324
- resistencias varias

2.3. ¿Qué debe saber antes del laboratorio?

2.4. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Introducción al amplificador operacional.
- Operación en modo diferencial y modo común.
- Amplificador operacional básico.
- Circuitos prácticos con amplificadores operacionales.

Después de haber hecho la práctica 1, estos temas corresponden a haber asistido a la clase teórica de la semana 1 (ver tabla 1) y/o haber leido las secciones 14.1, 14.3, 14.4 de la referencia [4] (Boylestad, sexta edición). Se sugiere también leer la hoja de datos del amplificador operacional LM324 [5] .

2.5. Introducción

[1]El amplificador operacional es un dispositivo que tiene como fin amplificar por un factor de ganancia G la diferencia entre dos señales de entrada.

$$OUT = G(IN_{(+)} - IN_{(-)}) \tag{2.1}$$

Un amplificador operacional idealmente tiene ganancia e impedancia de entrada infinita. La práctica se acerca bastante al modelo debido a que la impedancia de entrada es normalmente de decenas de megaohmios y la ganancia del orden de $10^5 \sim 10^7$.

En un circuito, el amplificador operacional se puede utilizar:

- En lazo abierto: Significa que no existe conexión entre la entrada en la salida, es decir, la entrada desconoce el resultado a la salida. El resultado de esto por lo general es la saturación del amplificador. Significa que debido a su alta ganancia cualquier diferencia por pequeña que sea,

[1]Tomado de [6].

va a producir una tensión muy alta a la salida, pero como el amplificador es un dispositivo activo, alimentado por fuentes de voltaje, en la práctica, el voltaje de salida solo aumente hasta un valor muy cercano al de la alimentación del amplificador.

- En lazo cerrado: Significa que existe una conexión o lazo que une la salida con la entrada y por lo tanto, existe una forma de conocer la salida para determinar un valor de entrada que se acerque más al objetivo. Es normalmente utilizado para realizar el control de variables eléctricas. La realimentación puede ser:
  - Positiva: Significa que la conexión se realiza con la terminal positiva, generalmente conduce a circuitos inestables o meta-estables utilizados en osciladores.
  - Negativa: Significa que la conexión se realiza con la terminal negativa, generalmente utilizada para conocer y actuar sobre variables de control.

Existen configuraciones comunes de amplificadores operacionales con un fin específico, de las cuales, algunas de estas son:

- Seguidor: En esta configuración se realiza una realimentación negativa, se utiliza como entrada la terminal positiva. El resultado es que la entrada es igual a la salida, sin embargo, la corriente que fluye en la entrada es cero (o casi cero, inferior a μA), por lo cual, los demás componentes colocados en la salida, no van a pedir corriente del circuito anterior.

- Inversor: En esta configuración se realiza una realimentación negativa, la terminal positiva es la referencia, entre la entrada y la salida del amplificador se coloca una resistencia R_1, y entre el voltaje de entrada y la entrada del amplificador otra resistencia R_2, de acuerdo a lo mostrado en la figura 2.1. El resultado es una red que invierte la polaridad del voltaje de entrada (+ a -,ó, - a +), y amplifica de acuerdo a la razón entre las resisténcias R_1 y R_2.

- No inversor: Equivalente al inversor, sin inversión de polaridad.

Como estos, existen más configuraciones para comparar, sumar, restar, integrar, etc. que puede consultar por ejemplo en el capítulo 14 de la referencia [4] (Boylestad, sexta edición).

Un caso especial es el amplificador de instrumentación el cual, permite variar la ganancia entre la entrada y la salida mediante resistencias que pueden ser internas o externas. Su funcionamiento es equivalente al de un amplificador en lazo abierto, sin la limitante de obtener valores solo de saturación. Además, varias de sus características son mejoradas, menores corrientes de entrada, mayor ancho de banda de funcionamiento, respuesta más rápida, etc.

2.6. Polarización del LM324

Para esta práctica utilizaremos el circuito operacional LM324 cuyo diagrama se muestra en la figura 2.2. Este circuito operacional puede alimentarse o polarizarse con un amplio rango de voltajes (0V a 32V ó -16V a +16V) entre sus terminales Vcc y tierra (terminales 4 y 11). Utilizaremos 5 voltios como fuente de alimentación para polarizar el LM324 ya que esta fuente es común a los circuitos digitales con los que eventualmente estos circuitos operacionales serán utilizados.

2.7. Procedimiento

1. Monte el LM324 [2] sobre el protoboard tal como se ilustra en la Figura 2.3 y use uno de los operacionales en lazo abierto para completar la tabla 2.1

$\mathbf{IN_{(+)}(V)}$	$\mathbf{IN_{(-)}(V)}$	**OUT(V)**
0	0	
0	5	
5	0	
5	5	
2.5	0	
0	2.5	

Tabla 2.1: Voltajes de salida de un amplificador operacional funcionando en lazo abierto.

[2] Hoja de datos [5].

2. Arme un circuito inversor para lograr una amplificación de 5 y 10 veces (aproximadamente). Compruebelo con una señal sinusoidal de 1 KHz. ¿Se recorta la señal de salida?. Explique. Use el DC offset para evitar el recorte de la señal de salida.

3. Arme un circuito no inversor para lograr una amplificación de 5 y 10 veces (aproximadamente). ¿Qué diferencias encuentra en el comportamiento de la señal de salida cuando se compara con el circuito inversor?.

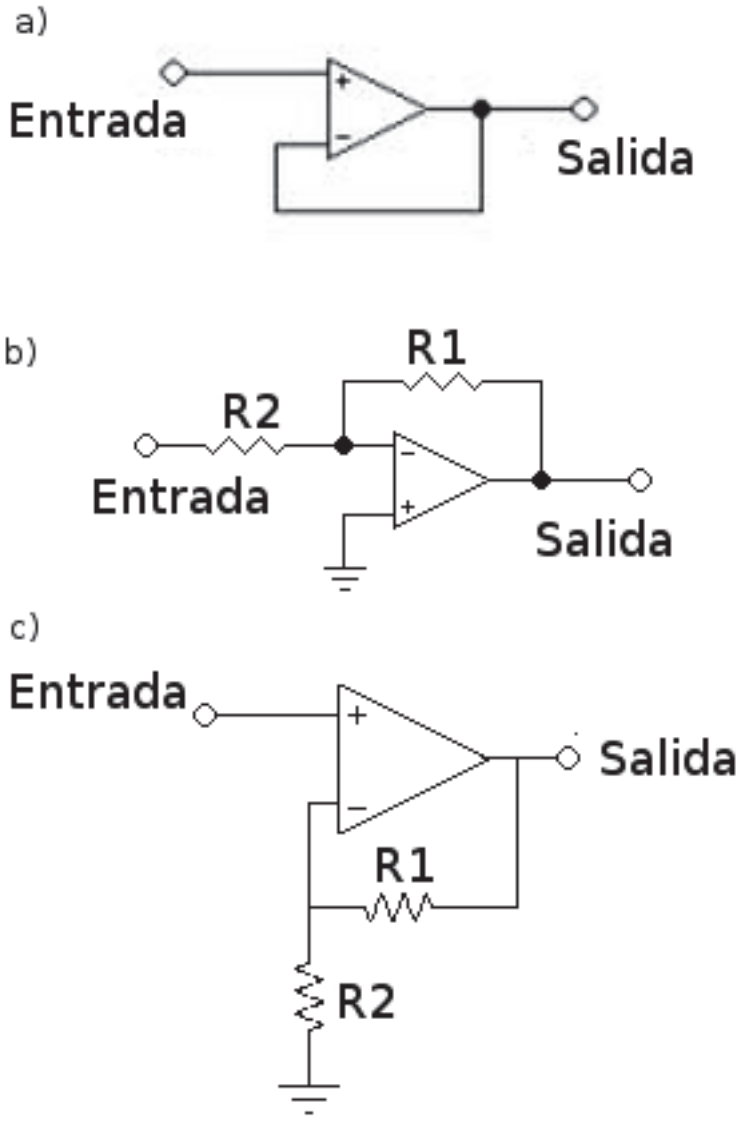


Figura 2.1: Amplificador operacional en modo a) seguidor, b)inversor, c) no inversor.

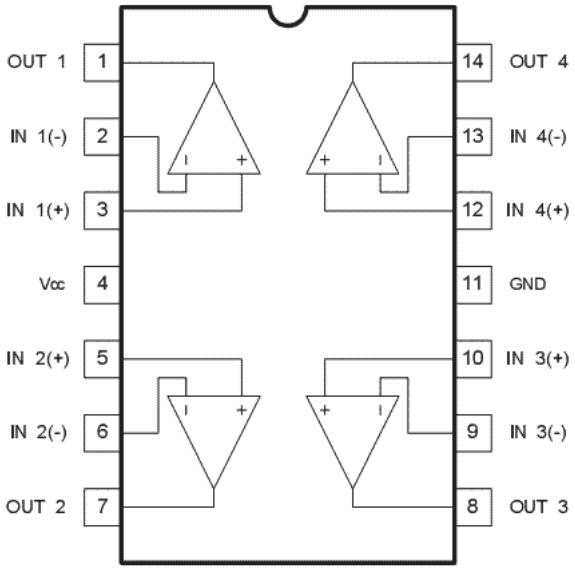


Figura 2.2: El amplificador operacional LM324.

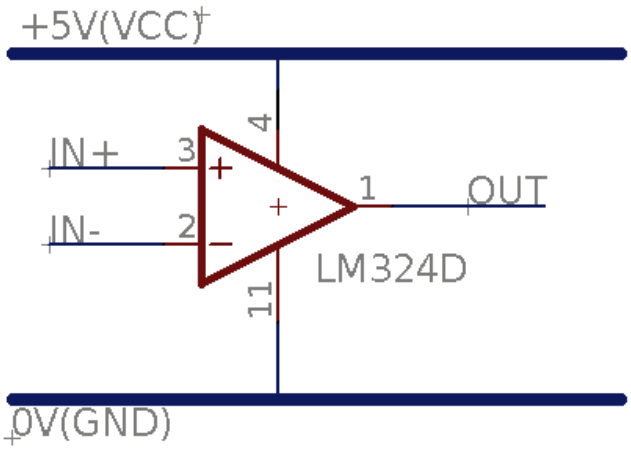


Figura 2.3: Montaje de un amplificador operacional LM324 en lazo abierto.

Práctica 3

Unidad Aritmética Lógica (ALU)

3.1. Objetivos

- Comprender el funcionamiento de una ALU.
- Diseñar y montar una ALU de 2 bits con los cubos logidule que se proporcionan en el laboratorio.

3.2. Materiales

- Cubos logidule (ver Apéndice C) con multiplexores, compuertas OR, XOR, AND, NAND. En caso de no contar con cubos logidule estos se pueden reemplazarse por compuertas comerciales de la serie 74xxx.

3.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Los números binarios.
- Conversiones de la base de números.
- Números octales y hexadecimales.

- Complementos.
- Operaciones lógicas.
- Compuertas lógicas digitales.
- Simplificación de productos de suma.
- Implementación con NOR y NAND.
- Lógica combinacional.
- Procedimientos de diseño.
- Sumadores.
- Dispositivos de gran y mediana escala de integración (LSI, MSI).
- Sumador binario paralelo.
- Decodificadores.
- Memoria de solo lectura (ROM).

Estos temas corresponden a haber asistido a las clases teóricas de las semanas 4 y 5 (ver tabla 1) y/o haber leido las secciones 1.2 a 1.5, 2.6, 2.7, 3.5, 3.6, 4.1 a 4.3, 5.1, 5.2 y 5.5 a 5.7 de la referencia [7] (Morris Mano, primera edición) y haber leido el Apéndice C de éste libro referente al uso de los cubos Logidule.

3.4. Introducción

La Unidad Aritmética Lógica (ALU) es el corazón de todo sistema computarizado. La ALU es un circuito combinacional MSI que realiza operaciones lógicas y aritméticas a dos operandos de entrada de acuerdo al estado de ciertas entradas digitales denominadas líneas de selección. En la Figura 3.1 se muestra el diagrama lógico de una ALU de 4 bits.

Que la ALU sea de 4 bits quiere decir que permite efectuar operaciones entre 2 números binarios (**A** y **B**) cada uno de 4 bits. Así, las conexiones A0 a A3 corresponden al primer operando o número binario de entrada **A** y en forma similar B0 a B3 corresponden al segundo operando o número binario

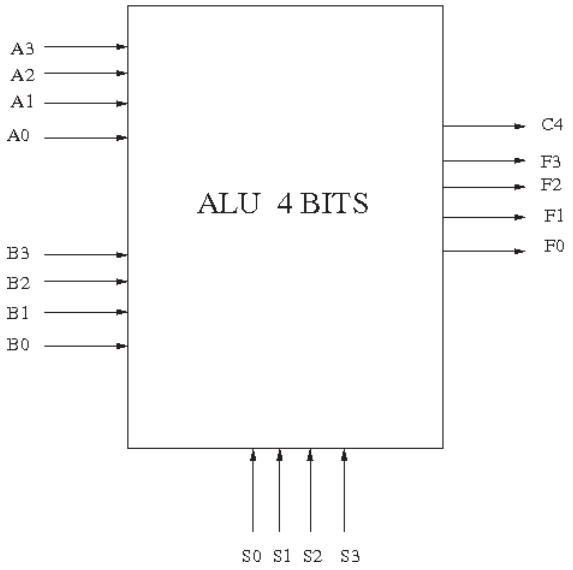


Figura 3.1: Diagrama Lógico de una ALU de 4 bits.

de entrada **B**. Las líneas S0 a S3 permiten elegir entre 16 posibles operaciones tal como se lista en la Tabla 3.1. El resultado de la operación se puede ver en los bits de salida F0 a F3 y para algunas operaciones se usa también el bit de salida C4 o carry.

3.5. Procedimiento

1. *a*) Sobre el diagrama lógico de la ALU de 4 bits escriba los valores binarios que deberían estar presentes en A0 a A3, B0 a B3 y S0 a S3 si se quisiera hacer la operación lógica **A OR B** donde, **A** = 1101 y **B** = 1011. Escriba también el resultado en F0 a F3 y C4.

   b) Sobre el diagrama lógico de la ALU de 4 bits escriba los valores binarios que deberían estar presentes en A0 a A3, B0 a B3 y S0 a S3 si se quisiera hacer la operación aritmética **A+B+1** donde **A** = 1101 y **B** = 1011. Escriba también el resultado en F0 a F3 y C4.

2. Una vez haya comprendido el funcionamiento de la ALU de 4 bits diseñe una ALU de solo 2 bits y que efectué las 6 operaciones indicadas en la

Líneas de Selección (S1 S2 S3 S4)	**Operación**	**Tipo de Operación**
0000	A	Lógica
0001	A OR B	Lógica
0010	A AND B	Lógica
0011	A OR B'	Lógica
0100	A AND B'	Lógica
0101	(A AND B)'	Lógica
0110	A'	Lógica
0111	B	Lógica
1000	B'	Lógica
1001	A XOR B	Lógica
1010	(A XOR B)'	Lógica
1011	A-B	Aritmética
1100	A+B+1	Aritmética
1101	A+B-1	Aritmética
1110	A-1	Aritmética
1111	B-1	Aritmética

Tabla 3.1: Lista ejemplos de operaciones posibles en una ALU.

Tabla 3.2.

Para el diseño de la ALU deberá seguir los siguientes pasos:

a) Diseñe y arme con los cubitos una ALU de 1 bit que realice solo las operaciones lógicas de la Tabla 3.2. Para ello debe utilizar un multiplexor, una compuerta OR, una compuerta AND, una compuerta XOR y una compuerta negadora (utilizar una NAND como negadora).

b) Haga el mismo diseño de 2*a*) pero ahora para 2 bits. Ármelo con los cubitos.

c) Para implementar la suma binaria (2 bits) necesita un sumador medio y uno completo. ¿Por qué? ¿Es posible obtener el sumador medio de las funciones que implementó en los puntos anteriores?

d) Teniendo en cuenta el punto 2*c*), diseñe y arme con los cubitos la funcion suma binaria (2 bits).

Líneas de Selección	**Operación**
000	F = A OR B
001	F = A AND B
010	F = A XOR B
011	F = A'
100	F = A + B (Suma binária)

Tabla 3.2: Operaciones para una ALU.

Práctica 4

Memoria de solo lectura (ROM)

Esta práctica no esta contenida en el cronograma mostrado en el prólogo de este libro pero es altamenete recomendada y se sugiere se asigne como tarea para realizar en la casa.

4.1. Objetivos

- Diseño de una memoria ROM de 8 palabras con longitud de palabra igual a 4 bits.
- Comprender el modo de almacenar información en una memoria y como recuperarla, proceso equivalente en memorias ROM y RAM.

4.2. Materiales

- 1 Dip Switch.
- Protoboard.
- Resistencias de 10 KΩ y 670 Ω.
- Condensadores de 0.1 μF y 0.33 μF.
- 8 leds pequeños.

- 1 decodificador 74LS138.
- Varios Integrados 74LS20.
- 1 Regulador Fijo de Voltaje de 5 V (LM7805).
- 1 Adaptador de voltaje fijo preferiblemente a 1 A (mínimo 8 V, máximo 20 V) ó un Adaptador de voltaje variable preferiblemente a 1 A.
- 1 jack para conectar adaptador al protoboard.
- Decodificador 74LS138.

4.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Los números binarios.
- Conversiones de la base de números.
- Números octales y hexadecimales.
- Complementos.
- Operaciones lógicas.
- Compuertas lógicas digitales.
- Simplificación de productos de suma.
- Implementación con NOR y NAND.
- Lógica combinacional.
- Procedimientos de diseño.
- Sumadores.
- Dispositivos de gran y mediana escala de integración (LSI, MSI).
- Sumador binario paralelo.

- Decodificadores.
- Memoria de solo lectura (ROM).

Estos temas corresponden a haber asistido a las clases teóricas de las semanas 4 y 5 (ver tabla 1) y/o haber leido las secciones 1.2 a 1.5, 2.6, 2.7, 2.5, 3.6, 4.1 a 4.3, 5.1, 5.2 y 5.5 a 5.7 de la referencia [7] (Morris Mano, primera edición) y haber leido la Sección B.3 referente al uso del protoboard. Tambien se sugiere haber leido las hojas de datos de los componentes 74LS138([8]) y 74LS20([9]). Este laboratorio se hará en la casa y se sugiere que el informe sea justo antes de realizar el laboratorio de lógica secuencial.

4.4. Introducción

Los dispositivos de memoria son utilizados para almacenar temporal o definitivamente datos que serán utilizados posteriormente en algún proceso. Dentro de los dispositivos electrónicos de almacenamiento magnético se distinguen dos clases de memoria: la memoria ROM y la memoria RAM.

- La memoria ROM o Read Only Memory es utilizada para almacenar la información de forma definitiva o con un largo periodo entre lecturas.
- La memoria RAM o Random Access Memory es utilizada para almacenar información que se utiliza con mayor frecuencia y que no necesita un almacenamiento definitivo.

Como resultado de su construcción y la evolución en la arquitectura de computadores, existen diferencias entre ellas en velocidad de acceso y tiempo medio entre fallas, sin embargo, la forma en la cual se accede a su contenido es similar.

Para ensamblar memorias ROM es común utilizar decodificadores. En esta práctica se usará el decodificador 74LS138[1] el cual es un decodificador de 3 entradas (A0 A1 A2) y 8 salidas **negadas** (O0 O1 O2 O3 O4 O5 O6 O7) lo cual está simbolizado por el pequeño circulito en cada una de las salidas, tal como se ilustra en el diagrama lógico del decodificador en las figuras 4.1 y 4.2[2].

[1] Hoja de datos [8].
[2] Imágenes tomadas del documento [8].

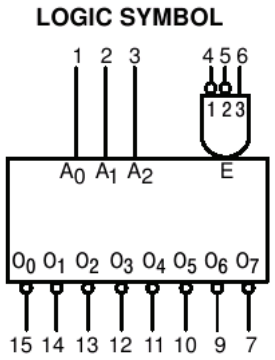


Figura 4.1: Diagrama esquemático del decodificador 74LS138.

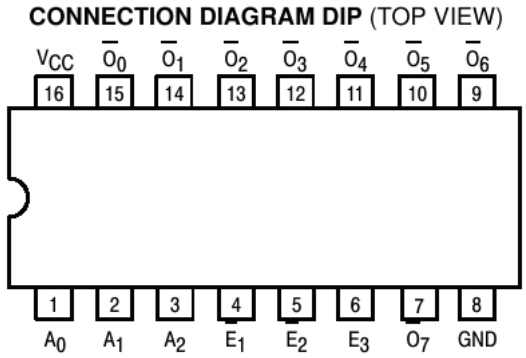


Figura 4.2: Distribución de pines del decodificador 74LS138.

Como consecuencia, la tabla de verdad del decodificador queda invertida; lo que era cero pasa a ser uno y viceversa, como se muestra en la Tabla 4.1[3]. Además de las entradas descritas, el 74LS138 tiene 3 entradas de habilitación E1 E2 y E3. Tal como lo indica el diagrama lógico del decodificador los valores que deben tener estas entradas son E1=0, E2=0 y E3=1. Cualquier combinación de valores diferentes deshabilitaría el decodificador, es decir, todas las salidas serían siempre uno sin importar el valor de las entradas A0 A1 A2.

[3]Tabla tomada del documento [8].

Entradas						Salidas							
E1	E2	E3	A0	A1	A2	O0	O1	O2	O3	O4	O5	O6	O7
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

Tabla 4.1: Tabla de verdad del decodificador 74LS138.

Si se quiere implentar una memoria ROM con este decodificador y compuertas OR y dado que las salidas del decodificador están negadas es necesario volver a negar todas las salidas lo cual puede aumentar en forma innecesaria el número de compuertas a utilizar. Una mejor alternativa es usar las salidas tal y como estan y unirlas con compuertas NAND en vez compuertas OR. Esto es posible ya que 2 salidas negadas, por ejemplo O1' y O2' alimentadas a una compuerta NAND salen de dicha compuerta como O1+O2 de acuerdo al teorema de Morgan. Para esta práctica se recomienda utilizar compuertas negadoras de 4 entradas como las contenidas en el circuito integrado 74LS20 (ver Figura 4.3).

4.5. Procedimiento

Implemente una memoria ROM de 8 x 4 usando un decodificador comercial de 3x8 como el 74LS138 y compuertas NAND de 4 entradas (e.g. las que se encuentran en el 74LS20[4]). "Queme" en la memoria su código de estudiante o el de su compañero (el que salga mas económico). Por ejemplo, si su código es 1132808 las palabras contenidas en su memoria ROM deberían

[4]Hoja de datos [9].

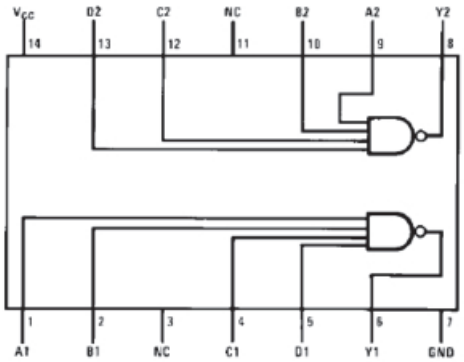


Figura 4.3: diagrama de pines de las compuertas NAND 74LS20.

ser como en la Tabla 4.2:

Dirección	**Palabra**
S2 S1 S0	F3 F2 F1 F0
000	0001
001	0001
010	0011
011	0010
100	1000
101	0000
111	1000

Tabla 4.2: Datos en la memoria correspondientes al código 1132808.

Simplifique al máximo para que su diseño tenga el menor número de compuertas posibles. Dibuje el circuito correspondiente.

4.6. Anexos

4.6.1. Conexión de los DIP Switch y los LEDs

El dip switch es el responsable de producir los unos y ceros que alimentan a las líneas de dirección S0 S1 S2 del decodificador. La Figura 4.4 muestra

como deben conectarse estos switches. Note que al cerrar cualquiera de los interruptores la respectiva linea (S0, S1, S2) se hace igual a 0 voltios (cero lógico), mientras que al abrirlo aparecerán 5 voltios en la línea respectiva correspondiente a un uno lógico. Los leds que se conectan a la salidas de la ROM (F3 F2 F1 F0) deben estar protegidos por una resistencia de 670 Ω.

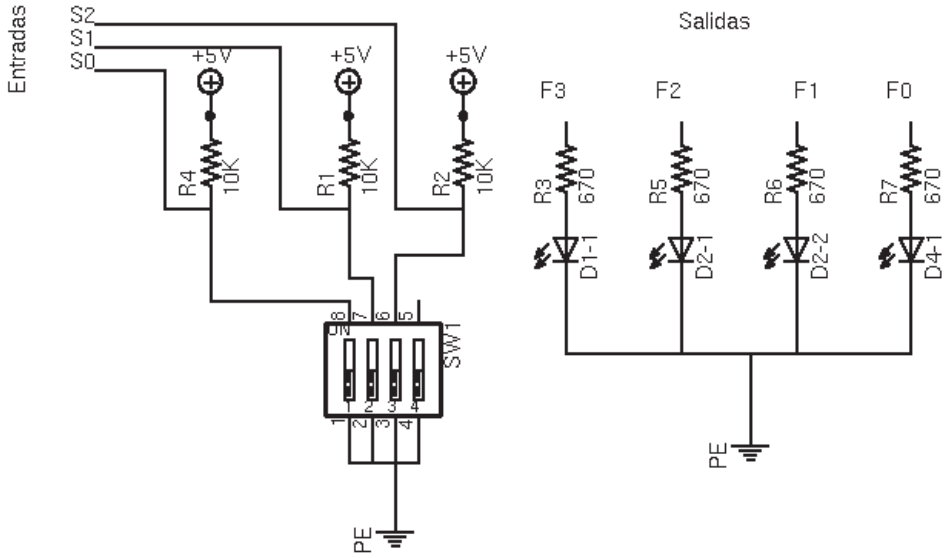


Figura 4.4: Conexión del dip switch.

4.6.2. Conexiones del regulador

La Figura 4.5 es el diagrama de conexiones del regulador 7805 cuya hoja de datos se puede encontrar en la referencia [10]. El adaptador de voltaje convierte el voltaje AC de la red eléctrica a voltaje DC y lo coloca en el pin 1 del regulador. Del pin 3 del integrado se obtienen +5V regulados. Tanto la entrada como la salida del regulador comparten el pin 2 que actua como línea común.

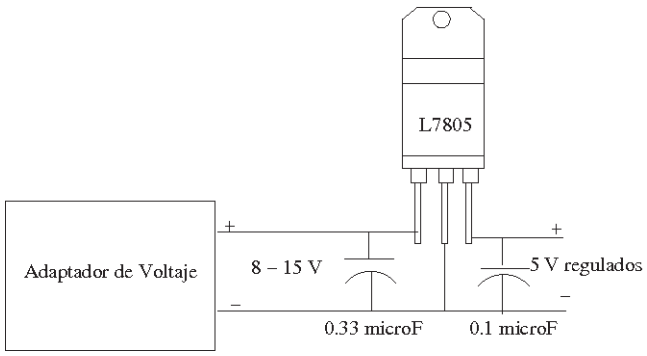


Figura 4.5: Diagrama ilustrativo de conexión del regulador de +5 V

Práctica 5

Lógica secuencial

5.1. Objetivos

- Aprender el funcionamiento básico de los flip-flops más utilizados (e.g. JK, T y D).
- Aprender las operaciones básicas que puede efectuar un registro (e.g. carga de datos en paralelo, carga de datos en serie, contador).

5.2. Materiales

- Módulos logidule con multiplexores.
- Compuertas lógicas.
- Flip-flops JK.

5.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Lógica secuencial síncrona.
- Flip-flops.
- Disparo del flip-flop.

- Análisis de circuitos secuenciales temporizados.
- Registros, registros de corrimiento.
- Contadores, contadores de ondulació o pulsación, contadores síncronos.
- Unidades de memoria, memorias de acceso aleatorio.

Estos temas corresponden a haber hecho las prácticas 3 y 4 y haber asistido a la clase teórica de la semana 7 (ver tabla 1) y/o haber leido las secciones 6.1 a 6.4, 7.1 a 7.5 y 7.7 a 7.8 de la referencia [7] (Morris Mano, primera edición).

5.4. Introducción

[1]Los flip-flops son circuitos oscilantes biestables que pueden permanecer en uno de estos estados por un tiempo indeterminado, es decir, osciladores que pueden tener solo dos valores, en este caso, '1' y '0', y que pueden mantener este valor. Debido a esta característica, son utilizados para almacenar información, que corresponde a la unidad mínima de almacenamiento, conocida como bit.

Los flip-flops tienen una salida (el dato en la memoria y/o su complemento) y uno o varios tipos de entrada, lo cual hace que existan dos tipos:

- Los flip-flops asíncronos (**latches**) solo tienen entradas que cambian directamente el valor de la salida. El más empleado es el latch RS.
- Los flip-flops sincrónicos (**flip-flop**) tienen una entrada adicional para que el cambio de valor se realice en sincronía con un reloj del sistema. La activación puede sincronizarse con el reloj:
  - En uno de los niveles, alto o bajo, es el caso de los flip-flop RS y D.
  - En uno de los flancos, de subida o de bajada, es el caso del flip-flop JK, T y D.

[1]Tomado de [11]

Los registros son la unión de varios elementos de almacenamiento (en este caso flip-flops). Cada flip-flop almacena un bit de información, así que la unión de n flip-flops, creará un registro de n bits. El tamaño más común para las arquitecturas de microcontroladores son registros de 8 bits (1 byte), lo cual, corresponde también en lenguaje C a una variable tipo `char`. Variables de mayor tamaño, e.g. `int` equivalente a 32 bits son sintetizadas por los microcontroladores como la unión de 4 registros, lo cual estará más claro al realizar la práctica 6 referente al manejo de PICs.

El interés de un registro es facilitar el manejo de datos, no solo bit por bit, sino un conjunto. Existen dos formas de realizar la carga y lectura de datos en esta memoria:

- Paralelo: cada flip-flop que compone el registro tiene sus entradas y salidas disponibles. Por lo cual, los bits se cargan o se leen de cada flip-flop independientemente. La ventaja es la rapidez, se pueden leer varios bits en un solo clock, la desventaja es el aumento del número de conexiones que incremente la posibilidad de daños.

- Serial: cada flip-flop tiene su entrada conectada a la salida del anterior. Los flip-flops de los extremos son utilizados para entrar datos (el que tiene las entradas libres), y leer datos (el que tiene la salida libre en el lado opuesto). La ventaja una reducción notable del número de conexiones, sin embargo, debido a que los bits salen uno por uno en cada clock, para leer un registro de n bits se necesita un tiempo de n clock para conocer su contenido completo.

Existen arquitecturas más complejas que combinan ambas posibilidades.

Las funciones comúnes y posibles sobre un registro son:

- Clear: Limpiar el registro (en general es colocar todos los bits en su valor inicial, para la mayoría de casos esto corresponde a cero, a menos que se especifique otro valor).

- Almacenar: Corresponde a colocar datos en el registro. Puede ser paralelo, serial o una combinación.

- Leer: Corrresponde a tomar los datos que fueron almacenados. Puede ser paralelo, serial o una combinación.

- Desplazar: Es muy común en los registros, tener que pasar los bits de un flip-flop a otro. Esto significa realizar un desplazamiento sobre los bits y existen dos posibilidades:

  - Desplazar a la derecha: pasar el bit más significativo a una posición menos significativa, e.g. el bit en la posición 7 a la posición 6, el de la posición 6 a la posición 5, etc.
  - Desplazar a la izquierda: pasar el bit menos significativo a una posición más significativa, e.g. el bit en la posición 2 a la posición 3, el de la posición 3 a la posición 4, etc.

  Con los bits de los extremos, depende del programa y/o la arquitectura que se use. Al desplazar a la derecha se observa que el bit 0 sale y el bit 7 queda vació. En algunos casos el bit que sale en 0 se inserta en el bit 0 (registro en anillo), en otros casos, el bit 7 va a otros sistemas y en el bit 0 se inserta un '0' lógico, etc. Lo mismo sucede con el desplazamiento a la izquierda.

5.5. Procedimiento

1. *a*) A partir del módulo JK dado para la práctica encuentre experimentalmente su tabla de verdad o tabla característica.

   b) Construya un flip-flop tipo D a partir del flip-flop JK dado y halle experimentalmente su tabla característica.

   c) Construya un flip-flop tipo T a partir del flip-flop JK y halle experimentalmente su tabla característica.

2. *a*) Usando un multiplexor de 2x1 (2 entradas, una salida, una línea de selección) , una compuerta negadora (o una NAND) y un flip-flop JK, dibuje (no hay que montarlo) el esquema de un circuito digital que convierta el JK en un flip-flop tipo D cuando la línea de selección del multiplexor sea cero y lo convierta en un flip-flop tipo T cuando la línea de selección sea uno.

   b) Verifique su funcionamiento en el papel (prueba de escritorio).

   c) Una vez dibujado este esquema, reemplace el multiplexor por las respectivas compuertas AND y OR que lo conforman. Halle la

función lógica que relaciona la línea de selección y la entrada del multiplexor con su salida. ¿Puede el multiplexor reemplazarse por una única compuerta?

d) Monte el circuito más sencillo que haya encontrado de los anteriores análisis con los cubitos y compruebe su funcionamiento.

3. Usando flip-flops JK, diseñe y monte un registro de 3 bits que permita cargar los datos en forma paralela, serial desde la izquierda y serial desde la derecha y que además sirva de contador. Puede usar como punto de partida la figura 5.1[2] y el resultado del punto anterior.

[2]Figuras tomadas de [7]

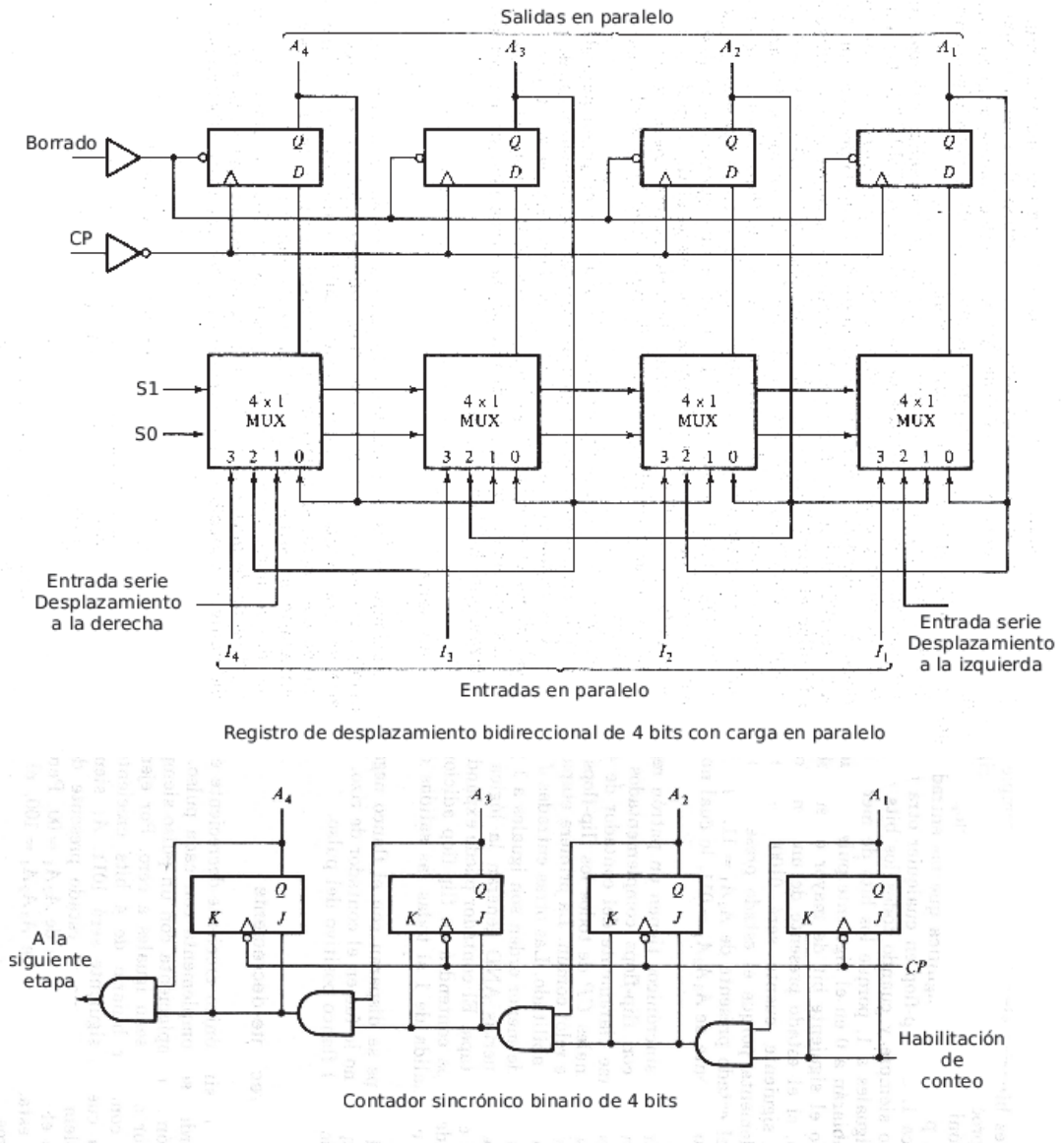


Figura 5.1: Ejemplo de registro de 3 bits.

Práctica 6

Manejo básico del PIC16F877

6.1. Objetivos

- Familiarizar al estudiante con los puertos de entrada/salida del PIC16f877 y las estructuras básicas de control de un programa para microcontroladores .
- Familiarizar al estudiante con el uso del PIC para solucionar problemas de lógica combinacional.

6.2. Materiales

- 1 Protoboard.
- 1 PIC 16f877-20/IP (OJO, NO COMPRAR EL PIC 16f877A).
- Resistencias de 10 KΩ, 1 KΩ, 220 Ω, 680 Ω.
- 8 Leds pequeños.
- 1 Adaptador de voltaje fijo preferiblemente a 1 amperio (mínimo 8 V, máximo 20 V).
- 1 Jack para conectar adaptador al protoboard.
- 1 Conector DB9 hembra para RS-232 con carcaza.
- 1 Condensador de 0.1 μF.

- 1 Condensador de 0.33 μF.
- 1 Dip switch de 8.
- 1 Cristal de 4 MHz.
- 2 Condensadores de 15$\sim$30 pF.
- 1 Pulsador de 2 patas.
- 4 Condensadores de 1 μF.
- 1 Integrado MAX232.
- 4 metros de cable rojo para protoboard.
- 4 metros de cable negro para protoboard.
- 8 metros de cable de colores surtidos para protoboard.
- 1 Caja para proteger el montaje.

Los cables que se van a soldar al DB9 deben ser de al menos un metro de largo y deben estar entorchados.

6.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Descripción del PIC 16F877.
- Organización de la memoria.
- Puertos de entrada y salida.
- Tipos de datos, operadores y expresiones en C.
- Control de flujo en C.
- Funciones y estructura de los programas en C.
- Características del compilador PICLITE.

- Librerías estandar, diferencias con ANSI-C, soporte para procesadores, formato de archivos de salida, macros, encabezados, configuraciones, identificación del dispositivo, eeprom.

- Instrucciones de un bit.

Estos temas corresponden a haber leido las secciones 1.0, 2.0 y 3.0 de la hoja de datos del PIC16F877 [12], haber leido los capítulos 2, 3 y 4 de la referencia [13] y haber leido las secciones 5.1 a 5.11 de la referencia [14] y/o haber asistido a la clase teórica de la semana 9 (ver tabla 1). También se recomienda haber hecho las prácticas 3, 4 y 5 y haber leido la hoja de datos del componente MAX232 [15].

El circuito que se muestra en la Figura 6.1 debe traerse armado para el día del laboratorio.

6.4. Introducción

[1] Los microcontroladores son circuitos integrados compuestos de:

- CPU: Unidad central de procesamiento o ALU. Es el lugar donde se realizan todas las operaciones.

- Memoria: Lugar de almacenamiento de la información.

- Puertos de entrada/salida: Lugares donde se realiza la comunicación con elementos externos.

Adicionalmente, la mayoría de microcontroladores cuentan ahora con periféricos que permiten llevar a cabo tareas especializadas sin interrumpir el procesamiento de la CPU.

Cuando se realiza un programa se están escribiendo instrucciones y datos que quedan guardadas en la memoria para su procesamiento por la CPU. Una de las características principales de un microcontrolador es la separación de la memoria de programa y la memoria de datos. Esta separación hace que cada vez que el programa se ejecute, el microcontrolador busque la siguiente instrucción en la memoria de programa y dependiendo de los datos que necesite para realizarla, se dirigirá a la memoria de datos.

[1] Tomado de [16].

La memoria de programa es no volátil, esto significa que aún si se suspende la alimentación eléctrica del dispostivo, los datos están aún almacenados para iniciar la próxima vez. En cambio la memoria de datos es volátil, una vez se apague el microcontrolador esta información se pierde y los valores de cada uno de los registros que componen la memoria típicamente se hacen cero.

La memoria de programa solo puede ser leída, no puede ser modificada por el funcionamiento del microcontrolador; para modificar el programa tiene que grabarse de nuevo la información, el microcontrolador **NO** puede modificar su propio programa. La memoria de datos si puede ser leida, grabada, limpiada, etc, tal como se hizo en la práctica 4 relativa a memorias ROM.

Una vez el microcontrolador ha sido programado y se inicia su funcionamiento, este inicia un ciclo de operaciones:

- Busca la primera instrucción y la deja disponible para procesarla en la ALU.
- Si la instrucción requiere datos adicionales, los busca en la memoria de programa o en la memoria de datos y los lleva a la ALU para ser operados.
- Realiza la operación en la ALU.
- El resultado es llevado a una posición en la memoria de datos, puede ser incluso la misma posición que tenía el dato de entrada, como es el caso de un incremento en uno, o a algunas de las entradas/salidas del componente.

La velocidad a la cual se realiza este proceso es característica del microcontrolador y varía entre fabricantes y arquitecturas, puede ir desde 12 ciclos de reloj para realizar todo el proceso de toma, procesamiento y resultado, hasta un solo ciclo de reloj para realizar todo.

Para el caso de este libro nos concentraremos en el PIC16F877, fabricado por la compañía Microchip Technology Inc.[2], con las siguientes características principales:[3]

- Frecuencia de reloj de 4 MHz.
- 4 ciclos de reloj por instrucción.

[2]http://www.microchip.com

[3]Datos tomados de la hoja de datos [12].

- Memoria de programa de 8K (palabras o registros de 14 bits).
- Memoria de datos 368 bytes.
- Memoria de datos EEPROM 256 bytes.
- 14 Interrupciones, se relaciona con procesos aleatorios a atender.
- Puertos A, B, C, D, E. Número de entradas/salidas disponibles.
- Arquitectura RISC, se relaciona con las operaciones disponibles en la ALU.
- In-Circuit-Serial-Progamming (ICSP), para programar sin retirar el componente.
- Alimentación de +5 V.
- Corriente de salida hasta 25 mA por pin.

Adicionalmente, contiene periféricos encargados de realizar funciones especiales para no realizar todo a través de la ALU, o para darle posibilidades adicionales:

- 3 Temporizadores , para llevar conteos de eventos o tiempo.
- 2 Modulos PWM , para realizar control sobre componentes externos.
- Comunicación Serial, MSSP y USART.
- Comunicación paralela, PSP.
- Conversor análogo-digital con 8 canales de entrada.

6.5. Preparativos

Para este laboratorio se utilizará el circuito mostrado en la Figura 6.1. Dicho circuito utiliza 2 pines del puerto C (RC7 y RC6) para “quemar” el PIC con los programas que se realicen y compilen en un computador debidamente configurado (la instalación de los paquetes y programas necesarios para realizar esta práctica se pueden encontrar en el Apéndice F). El computador

utiliza el RS232 para enviar este programa al PIC a través de un integrado MAX232[4] que convierte los niveles propios del RS232 (15 V y -15 V) a los niveles lógicos que soporta el pic (0 V y 5 V).

Para verificar que todo esta funcionando correctamente debe compilar y quemar el siguiente programa:

```
#include <pic.h>
__CONFIG(0x3d71);
void main(){
  TRISB = 0B00000000;  // Todos los bits como salida
  PORTB = 0B11111111;
  while(1);
}
```

Guarde este programa en un archivo llamado *prueba.c*. Compilelo utilizando el comando

```
$ compilar prueba.c
```

En la pantalla del computador se mostrará información acerca de donde es guardada la palabra de configuración, así como también del espacio de memoria que ocupará el programa (ignore por ahora los avisos de advertencia "Unknown prefix SPACEOPT").

Conecte el conector DB9 al puerto RS232 del computador. Conecte la fuente de alimentación al circuito del PIC. Si todo marchó bién en la etapa de compilación, se habrá creado el archivo *prueba.hex* que contiene el código de máquina a grabar en el PIC. Reinicie o resetee el PIC con el pulsador e inmediatamente después (menos de un segundo) use el comando

```
$ quemar prueba.hex
```

Si todo esta funcionado debidamente todos los leds del circuito del PIC deben encenderse. Si esto no sucede podría probar lo siguiente:

- ¿Están los leds invertidos?, recuerde que la pata mas larga del led debe conectarse al lado positivo.

- Revise el conector DB9. Las soldaduras en el conector no deben estar tocandose y el conector no debe haberse deformado después de haber hecho las soldaduras. Compruebe la conductividad en los cables soldados al conector.

[4]Hoja de datos [15].

- Pruebe los condensadores que se colocaron junto al cristal. Intercambielos por un par que usted sepa estén funcionando.
- Mida los voltajes en las patas de alimentación Vdd y Vss del PIC y tambien entre Vcc y GND del MAX232.
- Revise nuevamente el alambrado de su circuito. Con plano en mano cerciorese que todas las conexiones del PIC han sido alambradas correctamente.

6.6. Procedimiento

1. Una vez el circuito este funcionando correctamente edite el archivo fuente usando un número binario diferente en el puerto B, compile el programa y quemelo en el pic. Observe la correspondecia entre el nuevo número y los leds que se encienden en el protoboard.
2. Usando el mismo programa del paso 1 cambie PORTB=0B11111111 por PORTB =0x67. Compile y queme el nuevo programa en el PIC. ¿Qué leds se encedieron?. Convierta el número 0x67 a binario (procedimiento explícito por favor) y compare con los leds que se encendieron.
3. Usando el mismo programa del paso 1 cambie PORTB=0B11111111 por PORTB =123. Compile y queme el nuevo programa en el PIC. ¿Qué leds se encendieron?. Convierta el número 123 a binario (procedimiento explícito por favor) y compare con los leds que se encendieron.
4. Usando el mismo programa del paso 1 cambie PORTB=0B11111111 por PORTB =0163. Compile y queme el nuevo programa en el PIC. ¿Qué leds se encendieron?. Convierta el número 0163 a binario (procedimiento explícito por favor) y compare con los leds que se encendieron.
5. Usando el mismo programa del paso 1 cambie PORTB=0B11111111 por PORTB =-123. Compile y queme el nuevo programa en el PIC. ¿Qué leds se encendieron?. Convierta el número -123 a binario (procedimiento explícito por favor) y compare con los leds que se encendieron. Recuerde que el PIC representa los números negativos como complemento a 2 (que es lo mismo que complemento a 1 más uno). Todo el procedimiento debe escribirse en forma explicita.

6. Compile y queme en el PIC el siguiente programa:

```
#include <pic.h>
__CONFIG(0x3d71);
void main(){
  int i = 3245;
  TRISB =0B00000000;       // Todos los bits como salida
  PORTB = i ;
  while(1);
}
```

   Explique el resultado que obtiene en los leds. Puede ayudarse de la tabla de tipos de datos dada en clase (PORTB es una variable tipo char tal como se declara en */usr/hitech/include/pic1687x.h*).

7. Compile y queme en el PIC el siguiente programa:

```
#include <pic.h>
__CONFIG(0x3d71);
void main(){
  float  f = 0.21;
  TRISB =0B00000000;       // Todos los bits como salida
  PORTB = f*35;
  while(1);
}
```

   Explique el resultado que obtiene en los leds. Haga lo mismo con `f = 0.22`.

8. Compile, queme, compruebe y explique el funcionamiento del siguiente programa (ejemplo: estructura **if**).

```
#include <pic.h>
__CONFIG(0x3d71);
void main() {
  TRISB=0x00;//Todos los bits como salida
  TRISD=0xF0;//Cuatro bit mas significativos como entrada
  if(RD7 == 0){
```

```
        PORTB = 0x03;
    }
    else{
        PORTB = 0x0C;
    }
    while(1);
}
```

Elimine la linea `while(1);` y ejecute nuevamente el programa. ¿Qué diferencias hay?. Explique. Modifique el programa original de tal forma que se enciendan los bits RB0 y RB3 cuando RD7 = 0 y en caso contrario se enciendan los bits RB1 y RB2. El nuevo programa debe escribirse en el informe.

9. Compile, queme y compruebe el funcionamiento del siguiente programa (ejemplo: estructura **case**).

```
#include <pic.h>
__CONFIG(0x3d71);
void main(){
  int entrada;
  TRISB=0x00;       // Todos los bits como salida
  TRISD=0xC0;
  entrada = PORTD & 0xC0;
  switch(entrada){
      case 0x00 : PORTB = 0x04; break;
      case 0x40 : PORTB = 0x0A; break;
      case 0x80 : PORTB = 0x11; break;
      case 0xC0 : PORTB = 0x1F;
  }
  while(1);
}
```

Modifique el programa original de tal forma que cumpla con las condiciones mostradas en la Tabla 6.1.

10. El problema que se presenta en la práctica 9 puede ser resuelto por un circuito combinacional. Elabore la tabla de verdad y dibuje el correspondiente circuito. Resuelva el mismo problema utilizando el PIC en vez de lógica combinacional.

RD7=0 y RD6=0	RB0=1, los demás en cero
RD7=0 y RD6=1	RB1=1, los demás en cero
RD7=1 y RD6=0	RB2=1, los demás en cero
RD7=1 y RD6=1	RB3=1, los demás en cero

Tabla 6.1: Condiciones para el programa correspondiente al punto 9.

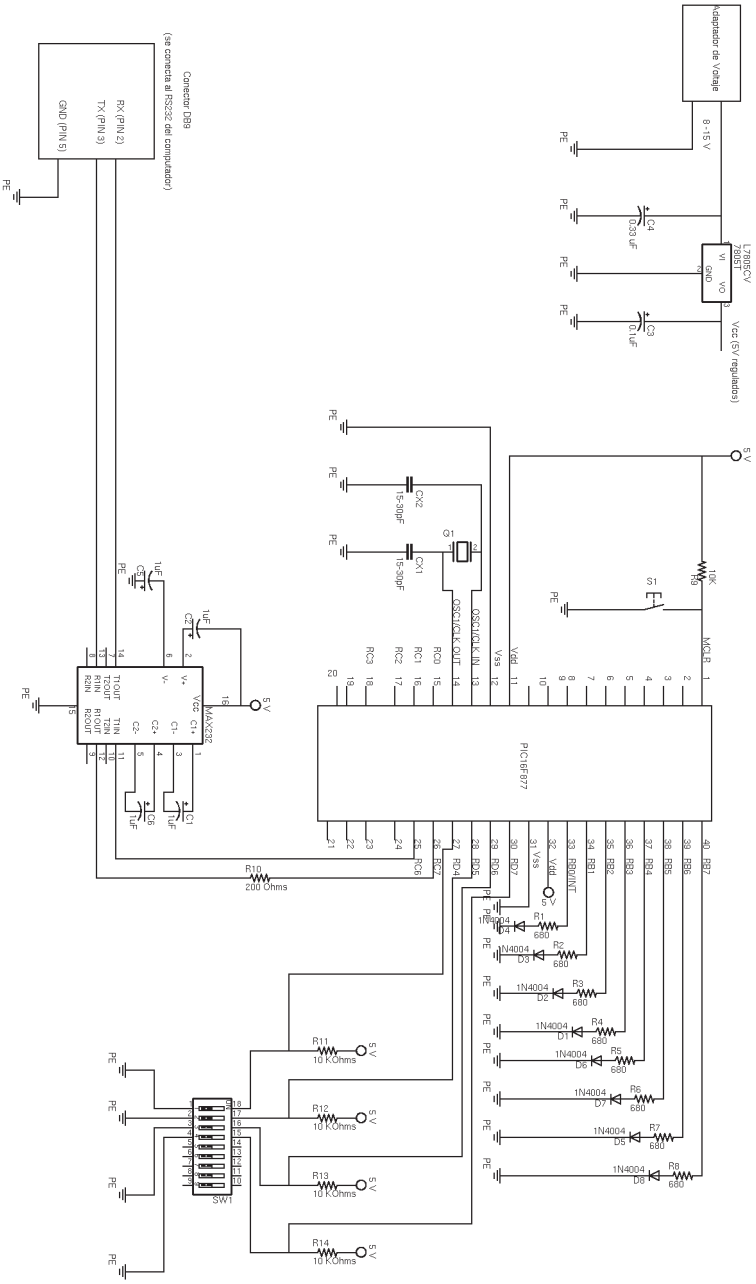


Figura 6.1: Circuito para los laboratorios iniciales con el PIC.

Práctica 7

Temporizadores del PIC16F877

7.1. Objetivos

- Introducir al estudiante a los temporizadores de un PIC cuando funcionan como contadores y como temporizadores.

7.2. Materiales

- Los mismos materiales que la práctica 6.

7.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Utilización del timer 0, 1 y 2

Estos temas corresponden a haber realizado la práctica 6 y haber leido la sección de temporizadores en el manual del PIC (secciones 5, 6 y 7 de [12]).

7.4. Introducción

El pic 16f877 tiene módulos especiales completamente independientes del microprocesador que pueden ser utilizados para hacer mediciones de tiempo, producir retardos de tiempo controlados o pueden hacer también conteo de eventos que sucedan fuera del pic. Dichos módulos reciben el nombre de

temporizadores y el 16f877 tiene 3 de ellos TMRO, TMR1 y TMR2. Todos ellos son básicamente contadores que pueden ser manejados por un reloj o evento externo en cuyo caso son llamados contadores o bién por el reloj del pic en cuyo caso son llamados temporizadores. En este laboratorio se usará el TMR2 para crear retardos controlados y el TMR0 para hacer conteo de eventos.

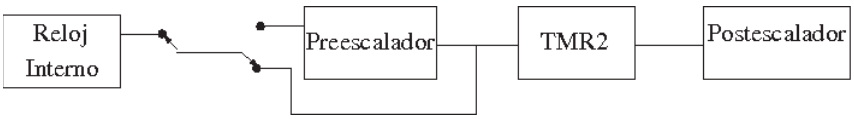


Figura 7.1: Diagrama funcional del timer TMR2.

7.4.1. Timer 2

EL TMR2 es un contador de 8 bits que posee tanto preescalador como postescalador. Estos circuitos son contadores que se conectan a la entrada de reloj del TMR2 como se ilustra en la Figura 7.1. Su función es dividir el pulso de reloj (sea este interno o externo) en el caso del preescalador entre 1, 4 y 16. Esto quiere decir, que si la frecuencia del reloj es de 1 MHz y se ha seleccionado el preescalador en 1 entonces hay conteo cada 1/1 MHz = 1 μs. Si se ha seleccionado 4 en el preescalador habrá conteo cada 4/1 MHz = 4 μs. El postescalador por su parte es un contador que cuenta cada vez que TMR2 alcanza su máximo valor de tal forma que si por ejemplo se selecciona 16 en el postescalador este genera una señal cada vez que el TMR2 ha contando 16 veces.

T2CON							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
b7	b6	b5	b4	b3	b2	b1	b0

R = Bit legible
W = Bit escribible
U = Bit sin implementar, al leer se obtiene '0'
-n = Valor cuando se ejecuta el POR Reset

Tabla 7.1: T2CON: Registro de control del Timer TMR2 (Dirección 12h)

La función que cumple cada bit en el registro **T2CON** se describe a continuación:

- Bit 7: **Sin implementar:** En la lectura se obtiene '0'.
- Bit 6∼3: **TOUTPS3:TOUTPS0:** Bits de selección de la post-escala de salida del Timer 2.
  - 0000 = Post-escala 1:1
  - 0001 = Post-escala 1:2
  - 0010 = Post-escala 1:3
  - ·
  - ·
  - ·
  - 1111 = Post-escala 1:16
- Bit 2: **TMR2ON:** Encendido del timer 2.
  - 1 = Timer 2 Encendido
  - 0 = Timer 2 Apagado
- Bit1∼0: **T2CKPS1:T2CKPS0:** Bits de selección de la pre-escala de entrada del Timer 2.
  - 00 = La pre-escala es 1
  - 01 = La pre-escala es 4
  - 10 = La pre-escala es 16

EL TMR2 usa al menos 6 registros para operar y ser configurado (ver manual del pic[12], pg. 55). Para este laboratorio solo usaremos tres de ellos: PR2, T2CON y TMR2.

PR2 es un registro de 8 bits y alli debe guardarse previamente el valor máximo que queremos que alcance TMR2 antes de reiniciarse.

T2CON (Tabla 7.1[1]) es un registro de configuración que sirve para establecer el valor del preescalador, el postescalador y para prender o apagar el

[1]Tomado de la hoja de datos [12].

TMR2. Por ejemplo si se almacena el valor 0x07 en T2CON estamos configurando el preescalador con 16, el postescalador en 1 y prendemos TMR2.

El registro TMR2 es el contador propiamente dicho y es allí donde queda almacenado el valor de la cuenta actual.

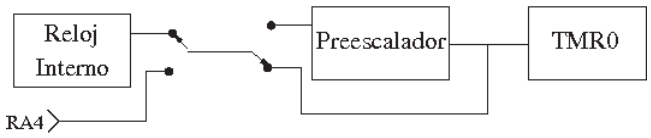


Figura 7.2: Diagrama funcional del timer TMR0.

7.4.2. Timer 0

EL TMR0 es un contador de 8 bit que puede ser configurado como temporizador o como contador. En este laboratorio TMR0 solo va a ser usado como contador y los eventos externos (pulsos) en este caso entrarán por el pin RA4 del pic (Figura 7.2). El TMR0 solo necesita 3 registros para operarlo y configurarlo **OPTION_REG**, **INTCON** y **TMRO**. EL registro **OPTION_REG** (Tabla 7.2) es el que configura TMR0. De acuerdo a los valores usados en este registro se puede escoger el valor del preescalador, si el TMR0 va actuar como temporizador ó como contador, si el evento externo activa el contador en el flanco de subida o de bajada, etc.

OPTION_REG							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU'	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
b7	b6	b5	b4	b3	b2	b1	b0

Bit legible
W = Bit escribible
U = Bit sin implementar, al leer se obtiene '0'
-n = Valor cuando se ejecuta el POR Reset

Tabla 7.2: OPTION_REG: Registro de control del Timer TMR0 (Dirección 81h, 181h).

La función que cumple cada bit en el registro **OPTION_REG** se describe a continuación:

- Bit 7 : **RBPU': Bit Habilitador de PULL-UP del Puerto B**

- 1 = Deshabilita los pull-ups del puerto B.
- 0 = Los pull-ups del puerto B son habilitados por registros individuales en cada puerto.

- Bit 6: **INTEDG:** Bit de selección del tipo de flanco de activación
  - 1 = Interrupción en el flanco de subida del pin RBO/INT
  - 0 = Interrupción en el flanco de bajada del pin RBO/INT
- Bit 5: **T0CS:** Bit de selección de la fuente de clock para TMR0
  - 1 = Transición en el pin RA4/T0CKI
  - 0 = Clock del ciclo de instrucciones internas (CLKOUT)
- Bit 4: **T0SE:** Bit de selección del tipo de flanco de activación de TMR0
  - 1 = Incrementese en una transición de alto a bajo en el pin RA4/T0CKI
  - 0 = Incrementese en una transición de bajo a alto en el pin RA4/T0CKI
- Bit 3: **PSA:** Bit de asignación del pre-escalador
  - 1 = Pre-escalador asignado al WDT
  - 0 = Pre-escalador asignado al modulo Timer0
- Bit 2: **PS2:PS0:** Bits de selección de la frecuencia del pre-escalador
  -

Valor del bit	Frecuencia TMR0	Frecuencia WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

7.5. Procedimiento

1. ```
   #include <pic.h>
   __CONFIG(0x3d71);
   void main(){
     TRISB=0;      // Todos los bits como salida
     PR2 = 250;    // Valor maximo de conteo para TMR2
     T2CON = 0x07; //prender TMR2 y configurar preescalador
     while(1){
       RB0 = 0;
       TMR2 = 0;
       while(TMR2 < 250); RB0 = 1;
       TMR2 = 0;
       while(TMR2 < 250); RB0 = 0;
     }
   }
   ```

   Este programa coloca en cero RB0, espera hasta que TMR2 cuente hasta 250 produciendo un retardo. Acto seguido, se enciende RB0 y nuevamente ocurre un retardo. A la salida de RB0 se va a ver una onda cuadrada cuya frecuencia va a depender del valor de PR2, la configuración del preescalador y de la frecuencia del reloj del pic. Compile y queme en el pic el anterior programa. Coloque la sonda de un osciloscopio en el pin RB0 del pic, observe y dibuje la señal que se produce. Tome nota de la frecuencia de esta señal o bien de su periodo. Si la frecuencia del reloj del pic es de 1 MHz calcule el periodo que la señal en RB0 debería tener. Tenga en cuenta que el preescalador se ha configurado en 16. ¿Coincide su cálculo con el periodo medido en el osciloscopio?. Repita el ejercicio cambiando 250 (en las tres líneas) por 100.

2. Repita el ejercicio 1 cambiando 250 (en las tres líneas) por 3 y T2CON = 0x04. ¿Qué sucede?. Tenga en cuenta que la ejecución de cada instrucción requiere de cierto tiempo. Para el pic 16f877 la mayoría de las instrucciones en código de máquina requieren de 1 μs para ser ejecutadas. Para ver las líneas de código de maquina generadas por su programa en c use el comando

   ```
   $ /usr/hitech/bin/picl -asmlist -16f877 archivo.c
   ```

   cuando compile el archivo fuente. Además del archivo *archivo.hex*, se
   ```

genera el archivo *archivo.lst*. Abra este último archivo y observe que tiene 4 columnas. La primera simplemente numera las líneas del archivo. La segunda es la dirección de la memoria flash donde se va a guardar la instrucción. La tercera instrucción es el código de máquina y la cuarta es la instrucción en assembler. Note que algunas intrucciones de su archivo fuente en c se traducen en varias instrucciones assembler (mire por ejemplo la instrucción TRISB = 0). Dado hay algunas instrucciones en código de máquina que se demoran 2 μs en vez de uno, es dificil hacer el cálculo exacto de los tiempos. Dibuje la forma de onda obtenida en el osciloscopio sobre la onda calculada (las diferencias pueden ser grandes). De una respuesta cualitativa a las posibles diferencias de acuerdo a lo discutido en este párrafo.

3. Si se desean producir retardos muy pequeños (del orden de los microsegundos) se puede utilizar la instrucción `NOP()` (No Operación) que se traduce en una sola instrucción de assembler y por lo tanto demora el PIC exactamente 1 μs. Mida el periodo en RB0 utilizando el siguiente programa:

```
#include <pic.h>
__CONFIG(0x3d71);

void main() {
  TRISB=0;       // Todos los bits como salida
  while(1){
    RB0 = 0;
    RB0 = 1;
  }
}
```

   Introduzca la instrucción `NOP()` en el programa anterior y mida nuevamente el periodo. Introduzca `NOP()` varias veces y mida nuevamente el periodo. ¿Qué puede concluir?.

4. El registro contador TMR2 solo puede contar hasta 255. Si se quieren producir retardos más grandes es necesario utilizar un ciclo que espere varios conteos completos de TMR2. Esto puede lograrse mediante ciclos **for** que encierren los comandos dentro de `while(TMR2 < 250)`.

Dado que estas líneas se repiten en 2 partes del programa principal es conveniente usar una subrutina como se ilustra a continuación.

```
 #include <pic.h>
__CONFIG(0x3d71);

void retardo(int in){
  int i;
  TMR2 = 0;
  for(i=0; i<in; i++){
    while(TMR2 < 250);
  }
}

void main() {
  TRISB=0;    // Todos los bits como salida
  PR2=250;    // Valor máximo de conteo para TMR2
  T2CON=0x07; // prender TMR2 y configurar preescalador
  while(1){
    PORTB=0;
    retardo(200);
    PORTB=255;
    retardo(200);
  }
}
```

Compile y queme en el pic este programa. Compruebe su funcionamiento con los leds colocados en el puerto B (deben prenderse y apagarse en aprox. 1 s). Modifique la subrutina de retardos de tal forma que usando `retardo(n)` se produzcan exactamente n segundos de retardo. Establezca el rango de valores de n para el cual es fiable la subrutina.

5. Escriba un programa para contar el número de eventos (pulsos) en el pin RA4 del pic. El número actual de pulsos debe poder visualizarse en el puerto B. Sugerencia: Utilice TMR0 en su modo de contador.

Práctica 8

Conversión Análogo-Digital (A/D)

8.1. Objetivos

- Familiarizar el estudiante con el uso del conversor A/D incluido en el PIC 16F877, su configuración y sus limitaciones
- Enseñar al estudiante el uso de librerías de comunicación para transmisión y recepción de datos entre el PIC y un PC.

8.2. Materiales

- Los mismos materiales utilizados en la práctica 6.
- 1 Potenciómetro de precisión de 5 KΩ.

8.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Métodos de conversión análoga digital.
- Resolución.
- Conversor análogo digital del PIC16F877.

Estos temas corresponden a haber leido el capítulo 5 de la referencia [17] y haber leido la sección 11 del manual del PIC[12] y/o haber asistido a la clase teórica de la semana 12 (ver tabla 1). También se requiere haber realizado las prácticas 6 y 7.

8.4. Introducción

El pic 16f877 tiene un conversor Análogo/Digital de 10 bits que funciona bajo el principio de aproximación sucesiva. El conversor posee dos registros de configuración ADCON0 y ADCON1 (Tablas 8.1 y 8.2[1]). El Conversor puede ser usado a través de cualquiera de las siguientes 8 patas del pic: 3 patas del puerto E y 5 patas del puerto A, tal como se ilustra en la Figura 8.1[2].

Para elegir el pin por la cual se quiere usar el conversor se usa el registro de configuración ADCON0 a través de sus bits CHS2, CHS1 y CHS0 (bits 5,4 y 3 respectivamente). Si por ejemplo, CHS2=0, CHS1=0 y CHS0 =0, la entrada al conversor A/D se hace a través de la pata RA0.

ADCON0							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE'	—	ADON
b7	b6	b5	b4	b3	b2	b1	b0

R = Bit legible
W = Bit escribible
U = Bit sin implementar, al leer se obtiene '0'
-n = Valor cuando se ejecuta el POR Reset

Tabla 8.1: ADCON0: Registro del AD (Dirección: 1Fh)

La función que cumple cada bit en el registro ADCON0 se describe a continuación:

- Bit 7∼6: **ADCS1:ADCS0: Bits de selección del clock de conversión del análogo-digital.**
  - 00 = $F_{osc}/2$
  - 01 = $F_{osc}/8$
  - 10 = $F_{osc}/32$

[1]Tablas tomadas de la hoja de datos [12].
[2]Figura tomada de la hoja de datos [12].

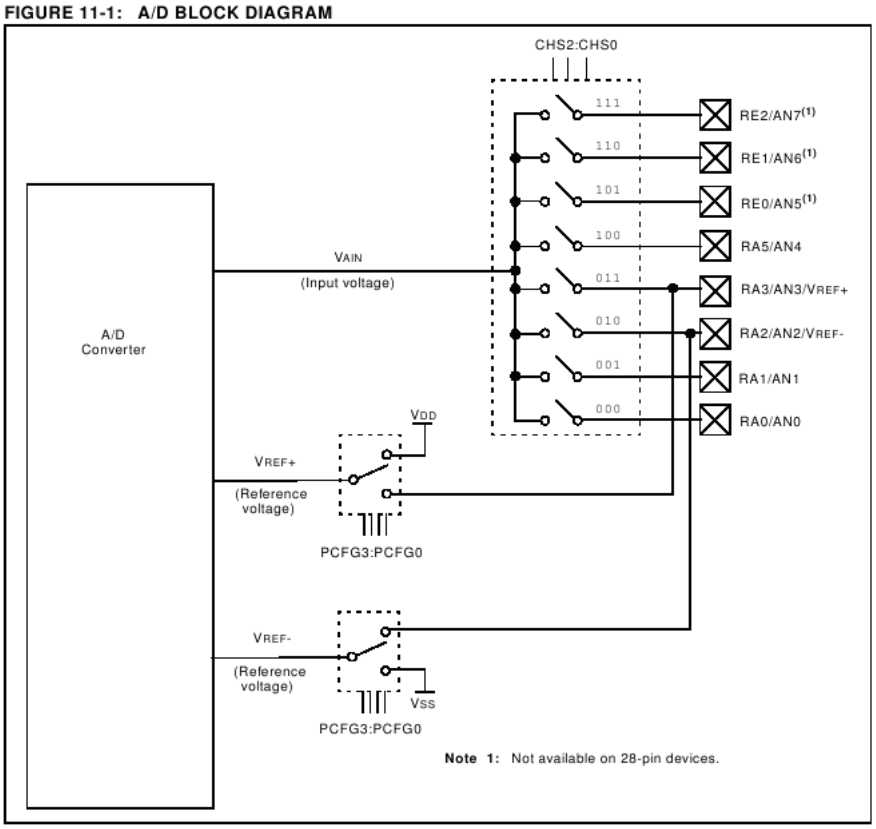


Figura 8.1: Diagrama funcional del conversor AD.

- 11 = F_{RC} (clock derivado de un oscilador RC)

- Bit 5∼3: **CHS2:CHS0:** Bits de selección del canal análogo.

  Nota 1: Estos canales no están disponibles en los dispositivos de 28 pines.

  - 000 = canal 0 (RA0/AN0)
  - 001 = canal 1 (RA1/AN1)
  - 010 = canal 2 (RA2/AN2)
  - 011 = canal 3 (RA3/AN3)
  - 100 = canal 4 (RA4/AN4)

ADCON1							
U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
b7	b6	b5	b4	b3	b2	b1	b0

R = Bit legible
W = Bit escribible
U = Bit sin implementar, al leer se obtiene '0'
-n = Valor cuando se ejecuta el POR Reset

Tabla 8.2: ADCON1: registro de control del conversor AD (Dirección: 9Fh)

- 101^1 = canal 5 (RA5/AN5)
- 110^1 = canal 6 (RA6/AN6)
- 111^1 = canal 7 (RA7/AN7)

- Bit 2: **GO/DONE':** Bit de estatus de conversión analogo-digital.

  Si ADON = 1

  - 1 = conversión analogo-digital en progreso (colocar en '1' este bit comienza la conversión)
  - 0 = conversión analogo-digital detenida (Este bit es colocado automáticamente en '0' por hardware cuando se completa la conversión analogo-digital)

- Bit 1: **Sin implementar:** Al leerse se obtiene '0'.

- Bit 0: **ADON:** Bit de encendido del conversor.

  - 1 = módulo de conversión encendido
  - 0 = módulo de conversión apagado y no consume corriente de operación

La función que cumple cada bit en el registro ADCON1 se describe a continuación:

- Bit 7: **ADFM: A/D Result format select**

  - 1 = Justificado a la derecha. Los 6 bits más significativos de ADRESH se leen como '0'

- 0 = Justificado a la izquierda. Los 6 bits menos significativos de ADRESL se leen como '0'

- Bit 6∼4: **Sin implementar**: Al leerse se obtiene '0'

- Bit 3∼0: **PCFG3:PCFG0:** Bits de control de la configuración del puerto A/D (Tabla 8.3).

PCFG3 PCFG0	AN7[1] RE2	AN6[1] RE1	AN5[1] RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	Vref+	Vref-	CHAN/ Refs[2]
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	Vref+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	Vref+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	Vref+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	Vref+	Vref-	A	A	RA3	VSS	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	Vref+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	Vref+	Vref-	A	A	RA3	RA2	4/2
1100	D	D	D	A	Vref+	Vref-	A	A	RA3	RA2	3/2
1101	D	D	D	D	Vref+	Vref-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	Vref+	Vref-	D	A	RA3	RA2	1/2

A = Entrada análoga
D = Entrada digital
Nota 1: Estos canales no están disponibles en los dispositivos de 28 pines
Nota 2: Esta columna indica el número de canales análogos disponibles como entradas A/D y el número de canales análogos usados como entrada de referencia de voltaje

Tabla 8.3: Configuración de los bits 3∼0 del registro ADCON1.

Como todo conversor A/D de aproximación sucesiva el conversor del 16f877 necesita 2 voltajes de referencia Vref+ y Vref-. Estos voltajes definen el rango de valores análogos que serán permitidos a la entrada del conversor. El pic 16f877 permite que dichos voltajes sean tomados directamente de Vdd y de Vss respectivamente ó bién que estos valores sean tomados del pin RA3 y el pin RA2. Todo esto es posible controlarlo a través del registro ADCON1 a través de sus bits PCFG3, PCFG2, PCFG1 y PCFG0.

La Tabla 8.2 que describe el registro ADCON1 muestra una tabla donde se puede ver de donde son tomados Vref+ y Vref- (columnas 10 y 11) para cada combinación posible de PCFG3, PCFG2, PCFG1 y PCFG0. Estos bits también determinan que patas de las 8 disponibles para entrar al conver-

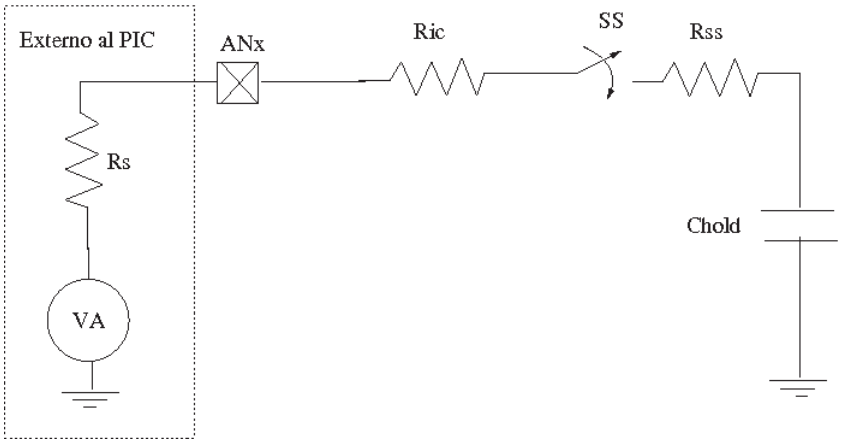


Figura 8.2: Diagrama esquemático del circuito muestreador.

sor van a funcionar como entradas análogas y cuales van a funcionar como entradas digitales.

El voltaje de referencia Vref+ puede estar ligeramente por encima de Vss. Exactamente Vss + 0.3. El voltaje de referencia Vref- mínimo puede llegar a ser Vss-0.3. La diferencia (Vref+ − Vref-) nunca puede ser menor a 2 voltios.

Justo en el momento de iniciar la conversión hay que tener en cuenta que al circuito de retención y muestreo del conversor (ver Figura 8.2) le toma cierto tiempo cargar el condensador C_{hold} donde finalmente se va almacenar el voltaje que quiere convertirse. Como se ve en la figura el circuito de retención y muestreo es en primera aproximación un circuito RC serie cuya constante de tiempo depende tanto de la resistencia interna (Rss + Ric) como de la resistencia equivalente (Rs) del voltaje que se quiere convertir.

Para una resistencia Rs=10 KΩ el tiempo de carga esta alrededor de los 20 μs. Este es el tiempo mínimo que hay que esperar para que el condensador se cargue. Dado que usualmente Rs esta por debajo de los 10 KΩ un tiempo de 20 μs es más que suficiente para cargar el condensador en la mayoría de los casos. En caso de requerir un tiempo de carga inferior se debe hacer el cálculo exacto con la fórmula de la sección 11.1 del manual del pic[12] (pág. 115).

Los tiempos de espera deben introducirse dentro del programa fuente que va a controlar la conversión A/D. No hay registros de configuración para esto.

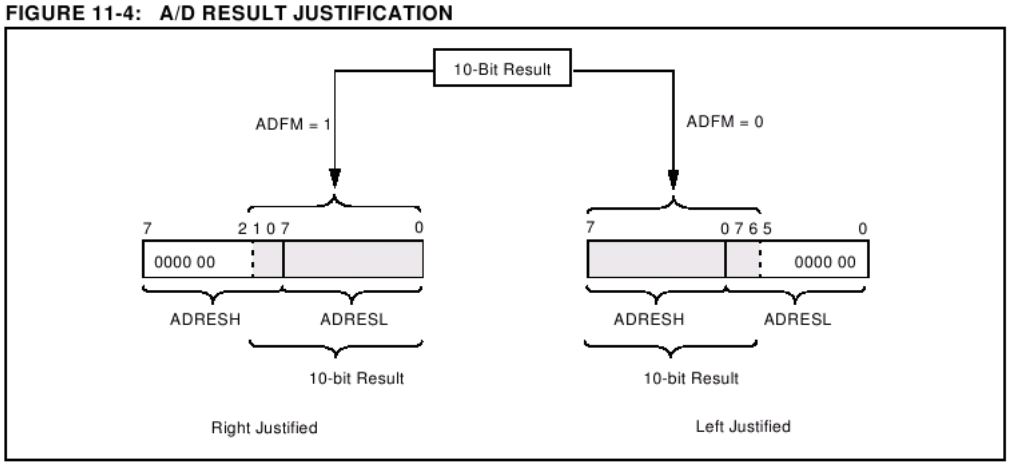


Figura 8.3: Ubicación del resultado de la conversión en los registros.

Otro aspecto a tener en cuenta es el tiempo "tad" que se demora el conversor en obtener un bit. El conversor de aproximación sucesiva es un circuito secuencial y por lo tanto necesita una señal de reloj. Dicha señal de reloj esta controlada por el cristal externo de 4 MHz que se coloca al circuito del pic. El tiempo de conversión "tad" se puede escoger en el registro de configuración ADCON0 en función del periodo de reloj Tosc=1/4 MHz= 0.25 μs. Esto se hace a través de los bits ADCS1 y ADCS0 del mencionado registro. Por ejemplo, si ADCS1=0 y ADCS0=0, tad=0.5 μs. De igual manera si ADCS1=0 y ADCS0=1, Tad=2 μs. La opción restante da Tad=8 μs. El tiempo tad debe ser mínimo de 1.6 μs de acuerdo a las especificaciones del fabricante. Así que en nuestro caso se debe configurar ADCON0 tal que tad sea de 2 μs o bién 8 μs.

El resultado de la conversión es guardado en 2 registros del PIC llamados ADRESH y ADRESL. Como el resultado de la conversión tiene 10 bits es posible guardarlos en los 16 bits de ADRESH y ADRESL justificados a la derecha o a la izquierda tal como se ilustra en la Figura 8.3[3]. Si se quiere el resultado de la conversión justificado a la izquierda se coloca el bit ADFM (bit 7) del registro ADCON1 en cero. Se coloca uno en este bit si el resultado se quiere justificado a la derecha.

[3]Figura tomada de la hoja de datos [12].

8.5. Envío de datos al PC

Los resultados de las conversiones A/D podrían visualizarse en los puertos del PIC a través de un conjunto de leds, un display de números de 7 segmentos (ver práctica 11) o bién directamente en la pantalla del computador. Con el circuito que estamos trabajando actualmente es relativamente fácil implementar la tercera opción. Para ello se utiliza el puerto serial del computador (RS232) que hasta ahora se ha usado solo para quemar los programas en el pic.

Para enviar bytes de pc a pic y viceversa usaremos la librería de comunicación que se describe en el apéndice G[4], en el directorio /usr/local/softwarepics/rs232-v6/. Se recomienda al estudiante que copie dicha librería en su directorio personal mediante el comando "cp -r /usr/local/softwarepics/rs232-v6/". Se creará el directorio rs232-v6/ donde se encuentran los programas que corren en el PC (tx1pc* tx2pc* y adpc*) y los que corren en el pic (tx1pic* tx2pic* y adpic*) con sus respectivas librerías (capser* y rs232*) y archivos de compilación.

Dado que ahora los programas incluyen librerias especiales ya no pueden ser compilados simplemente con el comando "compilar". Es necesario acudir al comando "make" y unos archivos especiales llamados "Makefiles" que compilan los programas incluyendo de forma automática las librerías que cada programa necesita. En todo caso el proceso de compilación se reduce a una sola línea de comandos. Por ejemplo, si queremos compilar el programa tx1pc.c usamos el comando "make -B -f Makefilepc PROGRAMA=tx1pc". Esto produce como resultado el archivo ejecutable tx1pc que puede correrse en el computador simplemente tecleando en una consola el comando "./tx1pc". Ahora, si lo que se quiere compilar es el programa del pic tx1pic.c se usa el comando "make -B -f Makefilepic PROGRAMA=tx1pic" lo cual produce el archivo tx1pic.hex que finalmente se quema en el pic con el comando usual. Noten que la única diferencia entre los comandos para compilar los programas del pic y los del pc (aparte del archivo mismo de programa) es el archivo Makefile. Las reglas y las librerías para compilar los programas del pic y los del pc son diferentes y por lo tanto necesitan archivos Makefile diferentes.

Tanto las librerías del pic como las librerías del pc proveen al programador

[4]Disponible en los computadores de la sala de instrumentación virtual del departamento de Física

con 2 instrucciones de comunicación: Una para leer lo que se envía a través de los 3 cables habilitados del puerto RS232 y otra para escribir información en dichos cables. Del lado del pic dichas instrucciones son sender() y receiver(). La instrucción sender(arreglo[], N) es la que escribe información en el puerto RS232 y tiene 2 argumentos. El primero es un arreglo tipo char que contiene N elementos. Una vez ejecutada esta instrucción, los N elementos del arreglo estarán disponibles en el puerto RS232 para que por ejemplo el computador o cualquier otro dispositivo que se encuentre conectado a dicho puerto los pueda leer. La instrucción del pic que lee lo que hay en el puerto RS232 es receiver(arreglo[], N). Cuando se ejecuta esta instrucción, el pic ESPERA hasta que se escriba información en el puerto RS232 por otro dispositivo como un computador u otro microcontrolador y coloca dicha información en los N diferentes elementos de la variable arreglo[]. De esta forma lo que se envió a través del RS232 es recuperado por el pic y guardado en la variable arreglo[] para ser usado después dentro del programa. El programa tx1pic.c es un claro ejemplo de como se utilizan las instrucciones receiver() y sender(). Del lado del computador las instrucciones para leer y escribir información en el RS232 son read() y write() respectivamente. El funcionamiento de estas instrucciones es muy parecido a las instrucciones receiver() y sender() del pic solo que las primeras tienen 3 argumentos en vez de 2 argumentos: el arreglo donde esta la información, el número de elementos del arreglo y un apuntador que indica en que puerto se lee o escribe información. Al igual que la instrucción receiver() del pic la instrucción read() ESPERA hasta que el otro dispositivo haya escrito algo en el puerto RS232. Este aspecto es clave para sincronizar el funcionamiento de los dos dispositivos que se están comunicando a través del RS232 y es la razón por la cual se enfatiza a lo largo de este párrafo. El programa tx1pc.c es un ejemplo claro de como deberían utilizarse las instrucciones read() y write(). En principio, lo explicado en este párrafo debería ser suficiente para realizar esta práctica. Sin embargo, si el estudiante desea profundizar en este tema se le recomienda revisar la práctica 14 y el apéndice G donde hay información detallada acerca de las librerías que estamos usando, algunos ejercicios, especificaciones del hardware, señales eléctricas y protocolos que efectivamente hacen posible la comunicación.

8.6. Procedimiento

1. Una vez instaladas las librerías de comunicación ubíquese en el directorio rs232-v6/ y desde allí compile (POR FAVOR LEA CUIDADOSAMENTE LOS PARRAFOS ANTERIORES PARA SABER COMO COMPILAR CON LIBRERÍAS) el programa tx1pic.c y quemelo en el pic. Compile el programa tx1pc.c y ejecutelo en el PC escribiendo en una consola ./tx1pc. Observe lo que sucede en la pantalla del computador y en los leds del puerto B del pic y descríbalo con sus propias palabras. Trate de entender la función de cada una de las líneas de código tanto en el programa del PC como en el programa del pic.

2. Compile el programa tx2pic.c y quemelo en el pic. Compile el programa tx2pc.c y ejecutelo en el PC escribiendo en una consola ./tx2pc. Ahora, se estan enviando paquetes de 2 bytes desde el PC al pic y de vuelta al PC. Observe y describa las principales diferencias entre estos programas y los programas tx1 que trabajaban con un solo byte. Escriba y pruebe un programa para el pic y otro para el PC que permita trabajar con paquetes de 4 bytes.

3. Coloque los extremos de un potenciómetro de precisión entre tierra y 5 voltios. Conecte la pata central del potenciómetro a la pata 2 (RA0/AN0) del pic 16f877. Compile y queme el programa adpic.c en el pic. Compile y ejecute el programa adpc.c en el computador y observe lo que sale en la pantalla. Mueva el tornillo del potenciómetro, resetee el pic y corra nuevamente el programa. Repita este procedimiento varias veces. Trate de entender cada una de las líneas de los programas que se usan tanto en el pic como en el PC. ¿Cuál sería el máximo valor y el mínimo valor que se verían en la variable conversion?. ¿A que valores de voltaje en AN0 corresponderían estos valores máximo y mínimo?.

4. Para los siguientes puntos es conveniente que el programa adpc tome datos en forma continua. Para logralo simplemente coloque dentro de un while(1) el bloque de instrucciones que empieza en n = write(fd, enviar, 1) y termina en printf (“Resultado completo de la conversion en formato decimal = %d\n \n”,conversion). También es posible que quiera activar el comando usleep(900000) para crear retardos de tiempo controlado dentro del programa del PC. Escoja tres o cuatro valores igualmente espaciados de Voltaje en AN0. Registre los correspondientes

números de la variable conversion. A partir de estos datos modifique el programa anterior para que la variable conversion muestre valores de 0 a 5 a medida que se cambia el voltaje en AN0 desde 0 voltios hasta 5 voltios. OJO: note que la variable conversión ya no puede ser del tipo entero.

5. Haga una curva de calibración, es decir, una gráfica del número mostrado por la variable conversión en el eje Y (después de la modificación) vs. el voltaje en A0 en el eje X medido con un voltímetro. Use al menos 20 datos que barran todos los valores desde 0 a 5 voltios.

6. Dibuje sobre la curva de calibración la curva ideal y = x. Dibuje en una gráfica aparte la diferencia de las dos curvas. ¿Qué diferencias hay entre la curva de calibración y la curva ideal ? ¿Qué tipos de errores estan presentes? ¿Sistemáticos, aleatorios o ambos?

7. Basado en la anterior gráfica que se puede decir acerca de la precisión y de la exactitud del voltímetro que acaba de construir con el pic. ¿Es posible calcular números para estos 2 parámetros? Si la respuesta es positiva, hágalo.

8. Teniendo en cuenta que el conversor A/D del pic es de 10 dígitos calcule la mínima resolución de su voltímetro. Según este cálculo, ¿Cuántos dígitos de la variable conversión son realmente confiables? Compare con lo obtenido en los puntos 6 y 7.

Parte II

Prácticas adicionales

Práctica 9

Lógica Combinacional

9.1. Objetivos

- Comprender el funcionamiento de las compuertas lógicas y el concepto de función lógica a través de un ejemplo concreto de aplicación de la lógica combinacional

9.2. Materiales

- Cubos logidule con funciones de compuertas lógicas variadas. En caso de no contar con cubos logidule estos se pueden reemplazarse por compuertas comerciales de la serie 74xxx.

9.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Los números binarios.
- Conversiones de la base de números.
- Números octales y hexadecimales.
- Complementos.
- Operaciones lógicas.

- Compuertas lógicas digitales.

Estos temas corresponden a haber asistido a las clase teórica de las semana 4 (ver tabla 1) y/o haber leido las secciones 1.2 a 1.5, 2.6, 2.7, 3.5 de la referencia [7] (Morris Mano, primera edición) y haber leido el Apéndice C de éste libro referente al uso de los cubos Logidule.

9.4. Introducción

Las compuertas lógicas son circuitos electrónicos que cumplen con las relaciones binarias establecidas por el álgebra de Boole, en la cual se le asignan valores númericos a los estados de verdad posibles. Es común utilizar, sin embargo, no es la única posibilidad:

- "Falso"='0'= 0 V
- "Verdadero = '1'= +5 V

Son válidas las operaciones de lógica básicas:

- NOT
- AND
- OR
- XOR

Cada una de estas compuertas se caracteríza por relacionar las entradas y la salida mediante una **tabla de verdad**. La combinación de compuertas en serie, paralelo o combinaciones lleva a relaciones más complicadas a la salida, sin embargo, todas se pueden representar mediante operaciones de suma de términos que se multiplican llamados **minterminos**, o multiplicación de términos que se suman llamados **maxterminos**. Cada una de estas posibilidades lleva a un mayor uso de compuertas AND (si usa minterminos) o de compuertas OR (si usa maxterminos).

Existe la posibilidad de reducir el número de compuertas a utilizar mediante operaciones algebraicas o diagramas de Karnaugh, las cuales pueden ser automatizadas mediante algoritmos. Pero la compuerta NAND que consiste en la conexión en serie de una compuerta AND y una NOT, tiene la

especial característica de permitir la obtención de cualquiera de las tablas de verdad básicas. Es por esto que los circuitos electrónicos (e.g. las memorias) se construyen con arreglos muy grandes de compuertas NAND que puedan llevar a cabo cualquier resultado mediante la configuración correcta. Su gran ventaja consiste en que un cambio en la lógica puede llevarse a cabo mediante cambios en las conexiones de la compuerta. La desventaja es que, en general, es necesario utilizar más compuertas (para lograr hacer una operación OR se necesitará de más de una NAND).

El mismo resultado se puede obtener con compuertas NOR, sin embargo, su utilización es mucho menos común.

9.5. Procedimiento

Algunos experimentos actuales de Física (e.g. en los aceleradores de partículas) requieren de campos magnéticos relativamente elevados. Usualmente esto se consigue haciendo circular grandes corrientes a través de bobinas que hacen las veces de electroimanes. Dichas corrientes producen calentamiento excesivo en los electroimanes por lo cual con frecuencia necesitan un sistema de refrigeración.

Se desea utilizar un sistema lógico para controlar de manera automática el tiempo que funciona uno de estos sistema de refrigeración. Sea W la salida de la operación lógica, donde 1 denota el hecho de encender el sistema de refrigeración y dejarlo así, y 0 la acción de apagarlo.

Las condiciones para determinar cuando poner en operación el sistema de refrigeración son las siguientes:

- El sistema de refrigeración trabaja de manera automática de las 7 a.m. a las 5 p.m. La variable T es 1 durante este lapso y 0 en el resto del día. Se hace uso de un dispositivo de temporización para proporcionar estas entradas al sistema, pero cualquier persona puede cambiar esta restricción manualmente.

- Cuando la temperatura ambiente baja por debajo de los 10 °C no es necesario utilizar el sistema de refrigeración. La variable R es 1 si la temperatura ambiente está por debajo de los 10 °C y 0 en caso contrario. Esta entrada será proporcionada por un detector de temperatura apropiado.

- El sistema debe ser capaz de trabajar manual o automáticamente. La variable M es 1 cuando el sistema está en operación manual y 0 cuando la operación es automática. Si la temperatura baja por debajo de los 10 °C mientras el sistema está en operación manual la refrigeración debe apagarse.

- También se dispone de un interruptor de potencia ON/OFF. La variable P es 1 cuando el sistema está en operación, ya sea manual o automática, y 0 cuando el sistema está apagado.

Sugerencia: Obtenga la tabla de verdad, obtenga la expresión booleana para W y finalmente monte el circuito correspondiente con los cubos suministrados en el laboratorio.

Práctica 10

LCD

Manejo de LCD (Liquid Crystal Display)

10.1. Objetivos

- Manejar un LCD de 4 dígitos y 7 segmentos usando un microcontrolador.
- Construir un cronómetro.

10.2. Materiales

- Los mismos materiales que en la práctica 6
- 1 LCD IM 50240.

10.3. ¿Qué debe saber antes del laboratorio?

Haber realizado las prácticas 6 y 7.

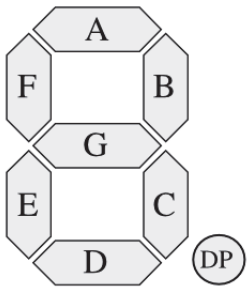


Figura 10.1: Display de 7 segmentos.

10.4. Introducción

10.4.1. Digitos de 7 segmentos

Para la realización de este laboratorio utilizaremos el LCD IM 50240 (OD8050) que usa los muy conocidos digitos de 7 segmentos. Estos dígitos constan de 7 segmentos rotulados con letras que van de la "a" a la "g", como se ilustra en la Figura 10.1. Para formar un 1 por ejemplo, se encienden el segmento b y el segmento c. Para formar un 2 se encienden los segmentos a, b, g, e y d y así sucesivamente. Es común usar un registro de 8 bits para almacenar el número a visualizar en el display. Dicho registro se conecta a cada uno de los 7 segmentos de tal forma que el segmento g quede conectado al bit menos significativo del registro, el segmento f quede conectado al siguiente bit y así sucesivamente (ver Tabla 10.1).

b07	b06	b05	b04	b03	b02	b01	b00
·	a	b	c	d	e	f	g

Tabla 10.1: Registro del display.

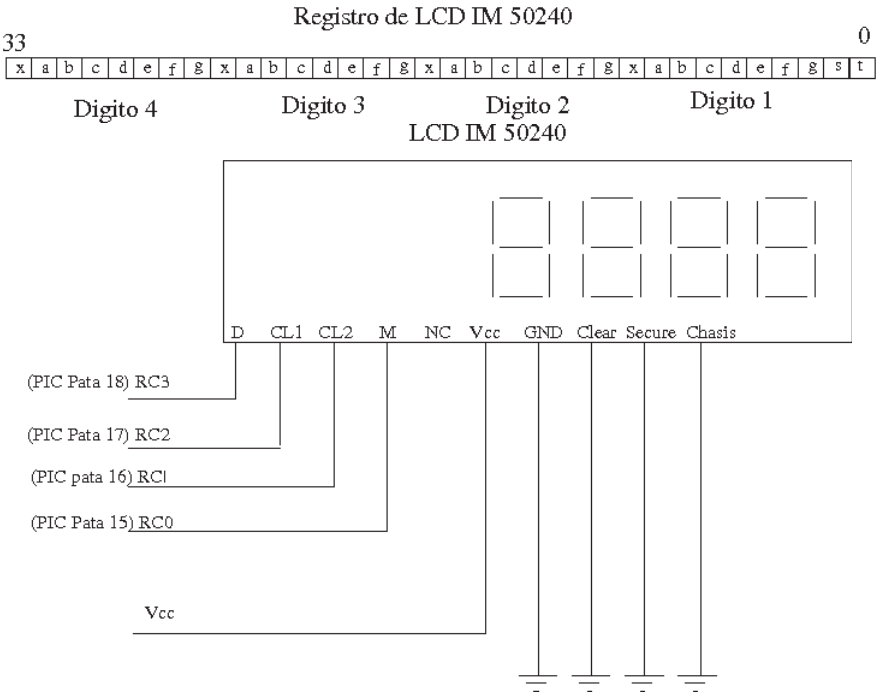


Figura 10.2: Registro del LCD.

10.5. LCD IM 50240

El LCD IM 50240 es un display de 4 digitos que usa un registro de 34 bits (ver Figura 10.2) donde alberga 4 subregistros de 8 bits cada uno para los 4 dígitos del LCD mas dos bits adicionales (s y t) para funciones especiales que siempre se deben cargar con ceros. El LCD IM 50240 carga el registro de 34 bit en forma serial y usando desplazamiento a la derecha. En otra palabras los datos entran por el bit 33 a través del pin D y se van desplazando hacia la derecha hasta que finalmente después de 34 desplazamientos el registro queda cargado. Para realizar el desplazamiento a la derecha es necesario enviar un pulso por el pin CL2. El desplazamiento ocurre en el flanco de bajada de dicho pulso. Una vez que el registro esta cargado con sus 34 bits es necesario enviar un pulso por CL1 para que se muestre el contenido del registro en el display. Note que para cambiar cualquier dígito es necesario cargar todo el registro otra vez.

Pin	Función	Descripcion
1	D	entrada de datos
2	CL1	muestra dato actual del registros de datos
3	CL2	señal de desplazamiento a la derecha
4	M	refresca la pantalla
5	N.C.	no conectar
6	Vcc	alimentación
7	GND	tierra
8	CLEAR	pulsador, no afecta los registros del LCD
9	SECURE	pulsador, no afecta los registros del LCD
10	CHASIS	aterrizar el marco para evitar estática

Tabla 10.2: Señales para el LCD.

10.6. Procedimiento

Para realizar los ejercicios de esta práctica debe añadir el LCD IM 50240 a su circuito original del PIC (Figura 6.1) tal como se ilustra en la Figura 10.2.

1. La siguiente subrutina permite enviar digitos decimales (0 a 9) a cada uno de los dígitos del LCD IM 50240.

```
//Muestra los números colocados en la funcion display
#include <pic.h>
__CONFIG(0x3d71);    // Cristal externo
#define D   RC3
#define CL1 RC2
#define CL2 RC1
#define M   RC0
void display(char dig4, char dig3, char dig2,char dig1){
  char numero[10],dig[4], i, j;
  dig[0] = dig1;
  dig[1] = dig2;
  dig[2] = dig3;
  dig[3] = dig4;
  TRISC = 0;
  D = 0;
  CL2 = 1; CL2 = 0;
```

```
  CL2 = 1; CL2 = 0;
  for(j=0; j<=3;j++){
    switch(dig[j]){
      case 0x00 : numero[j] = 0x7E; break;
      case 0x01 : numero[j] = 0x30; break;
      case 0x02 : numero[j] = 0x6D; break;
      case 0x03 : numero[j] = 0x79; break;
      case 0x04 : numero[j] = 0x33; break;
      case 0x05 : numero[j] = 0x5B; break;
      case 0x06 : numero[j] = 0x5F; break;
      case 0x07 : numero[j] = 0x70; break;
      case 0x08 : numero[j] = 0x7F; break;
      case 0x09 : numero[j] = 0x73;
    }
    for(i = 0; i<8; i++){
      D=numero[j]&1;
      numero[j] = numero[j] >> 1;
      CL2=1;CL2=0;
    }
  }
  CL1 = 1; CL1 = 0;
  M=M^1;
}

void main(){
    display(1,2,3,4);
}
```

Compile, queme este programa en el pic y compruebe su funcionamiento. Si todo esta funcionando correctamente debe verse la secuencia “1” “2” “3” “4” en el LCD. Trate con secuencias de numeros diferentes. Estudie cuidadosamente el programa y coloque comentarios adecuados en cada una de sus líneas.

2. Modifique la subrutina del punto 1 de tal forma que sea posible visualizar digitos hexadecimales, es decir además de los dígitos decimales se puedan visualizar las letras de la “a” a la “f” como se ilustra en la Figura 10.3. Recuerde que para distinguir variables en C de simples

Figura 10.3: Segmentos a encender para formar letras.

letras o caracteres se deben utilizar comillas simples. Por ejemplo, la línea e lenguaje C,

```
i = 'a'
```

significa que la variable i queda cargada con el valor ascii de la letra o el caracter a.

3. Utilizando el ejercicio 4 del práctica 7 y la subrutina para el display del punto 1, diseñe un cronómetro con décimas de segundos, que alcance un máximo de 9 minutos, 59 segundos y 9 décimas de segundo.

   El cronómetro debe tener 2 botones. Uno de ellos para colocar el cronómetro en ceros y el otro para iniciarlo y pararlo. Para los botones o pulsadores se recomienda usar los pines libres del puerto C ya que sus entradas son ST (Schmidtt trigger).

Práctica 11

Conversión Análogo-Digital (A/D) con visualización en LCD

11.1. Objetivos

- Familiarizar el estudiante con el uso del conversor A/D incluido en el PIC 16F877, su configuración y sus limitaciones.
- Construcción de un voltímetro de 4 dígitos.

11.2. Materiales

- Los mismos materiales utilizados en la práctica 6.
- 1 Potenciómetro de precisión de 5 KΩ.
- 1 LCD IM 50240..

11.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Métodos de conversión análoga digital.

- Resolución.
- Conversor análogo digital del PIC16F877.

Estos temas corresponden a haber leido el capítulo 5 de la referencia [17] y haber leido la sección 11 del manual del PIC[12] y/o haber asistido a la clase teórica de la semana 12 (ver tabla 1). También se requiere haber realizado las prácticas 6, 7 y 10.

11.4. Introducción

Ver Introducción de la sección 8.4.

11.5. Procedimiento

A diferencia de la práctica 8 en esta práctica vamos a utilizar el LCD de la práctica 10 para visualizar los resultados del conversor de 10 bits del pic 16f877.

Este programa muestra el valor del registro ADRESH y ADRESL en el LCD en formato decimal. (Es decir que barriendo 0 a 5 voltios en la entrada A0 se obtiene un número de 0 a 1023 en el LCD).

```
void main() {
    int temp;
    char dig4, dig3, dig2, dig1;
    ADCON1=0x80;//resultado justificado a la derecha
    ADCON0=0x41;//selecciono A0 como entrada análoga(pata 2)
    PR2 = 40;   // máximo conteo de TMR2
    T2CON = 0x04;// enciende timer 2
    TRISC = 0;   // PORTC como salida
    TRISB = 0;   // PORTB como salida
    while(1){
      TMR2 = 0;
      while(TMR2<40); ADGO = 1;
           //espera al menos 40 microsegundos
           //para cargar el condensador de muestreo
           //(20 microsegundos ) y hacer la conversión
           //(10 bits * 2 microsegundos)
```

```
      while(ADGO==1); PORTB =ADRESL;
      temp = 256*ADRESH+ADRESL;
      dig4 = temp/1000;
      dig3 = (temp - dig4*1000)/100;
      dig2 = (temp -dig4*1000 -dig3*100)/10;
      dig1 = temp -dig4*1000 -dig3*100 -dig2*10;
      display(dig4,dig3,dig2,dig1);
    }
}
```

Compile y queme el anterior programa en el pic. Note que se usa la subrutina "display" que se estudio en la práctica 10 y por lo tanto debe añadirse antes del programa principal tal y como se hizo en la mencionada práctica

1. Escoja tres o cuatro valores igualmente espaciados de voltaje en A0. Registre los correspondientes números mostrados en la pantalla del LCD. A partir de estos datos modifique el programa anterior para que el LCD muestre valores de 0.000 a 5.000 a medida que se cambia el voltaje en A0 desde 0 V hasta 5 V.

2. Haga una curva de calibración, es decir, una gráfica del número mostrado en el LCD en el eje Y (después de la modificación) vs el voltaje en A0 en el eje X. Use al menos 20 datos que barran todos los valores desde 0 a 5 V.

3. Dibuje sobre la curva de calibración la curva ideal y = x. Dibuje en una gráfica aparte la diferencia de las dos curvas. ¿Qué diferencias hay entre la curva de calibración y la curva ideal? ¿Qué tipos de errores estan presentes ? ¿Sistemáticos, aleatorios o ambos?

4. Basado en la anterior gráfica que se puede decir acerca de la precisión y de la exactitud del voltímetro que acaba de construir con el pic. ¿Es posible calcular números para estos 2 parámetros?. Si la respuesta es positiva, hagalo.

5. Teniendo en cuenta que el conversor A/D del pic es de 10 dígitos calcule la mínima resolución de su voltímetro. Según este cálculo, ¿Cuántos dígitos del LCD son realmente confiables?. Compare este último resultado con lo obtenido en los puntos 3 y 4.

Práctica 12

Las interrupciones

12.1. Objetivos

- Usar las interrupciones para atender procesos externos del PIC y para atender llamados del hardware extra del pic como temporizadores, conversor, etc

12.2. Materiales

- Los mismos materiales utilizados en la práctica 6.

12.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Manejo de interrupciones en el PIC16F877.

Estos temas corresponden a haber realizado las prácticas 6 y 7, retomando la sección de manejo de interrupciones.

12.4. Introducción

Cuando es necesario leer señales externas al microcontrolador o señales de funciones especiales como por ejemplo contadores el uso de interrupciones es la forma más eficiente de realizar esta tarea. Se relacionan con eventos que

suceden en cualquier momento y no necesariamente están relacionados con una secuencia lógica en el programa principal.

Una interrupción en un microcontrolador es una llamada de atención, que indica que algo debe ser atendido, por lo cual la ejecución del programa principal se detiene momentáneamente, se almacena el punto en el cual se quedó y se procede a revisar la naturaleza de la interrupción.

Generalmente las fuentes de interrupción son elementos periféricos a la CPU del pic, es por tal razón que para el PIC16F877 existen 14 fuentes diferentes de interrupción: cambio en uno de los pines, cambio en los niveles de entrada, conversor análogo-digital, USART, etc. Cada una genera sus señales de interrupción para ser atendidas por el procesador, y es el programa el que decide que interrupciones atender, en que casos y que hacer.

Cada interrupción tiene un habilitador (Enable bit) y todas tienen un bit de habilitación en común. Es por esto que si se quiere habilitar las interrupciones de un módulo, es necesario habilitar las interrupciones en general con el bit GIE del registro INTCON, y por ejemplo vamos a habilitar las interrupciones cuando se genere un cambio en el pin RB0, habilitando el bit INTE del mismo registro. Debido a que las demás interrupciones no se habilitaron, no importa si los otros módulos producen o no señales para ser atendidos, estos simplemente no serán tenidos en cuenta.

Cuando se genera una fuente de interrupción, el pic detiene las operaciones del programa que ejecuta y guarda su estado en bancos de memoria. Se dirige entonces a un sector de memoria especializado donde se encuentran las instrucciones a ejecutar en esta situación conocido como **vector de interrupciones**[1]. Luego de terminar con la ejecución, el programa retorna a su flujo normal.

Debido a que no existe un punto secuencial en donde aparezca la atención de interrupciones dentro del programa, es necesario crearlo como una función adicional con el nombre **interrupt** que el compilador interpretará especialmente para ubicarla en las posiciones de memoria correctas. Debido a que esto no hace parte de la utilización standard del lenguage C, la forma de implementación, la estructura y el lenguaje dependen enteramente del compilador y en este caso se utilizará `picl`. En caso de necesitar detalles, es necesario remitirse al manual del compilador [18].

[1]En general, el vector de interrupciones no contiene las instrucciones ya que algunas interrupciones pueden llevar muchos pasos para ser atendidas. Lo que normalmente es almacenado en el vector de interrupciones es la dirección de memoria donde inician las instrucciones que atienden la interrupción

Si un periférico genera una interrupción y esta está habilitada, entonces por hardware se pone en alto una bandera que identifica que periférico necesita ser atendido (en este caso son bits llamados Flags en los registros de configuración de las interrupciones). En caso de presentarse varias interrupciones a la vez, se atienden en orden jerárquico determinado por la architectura del pic. En la mayoría de casos cuando se atiende una interrupción es necesario bajar la bandera (escribir un cero en ese bit), de lo contrario, no se sabrá si ya fue atendida o no, lo cual es problemático en caso de conteos o lectura de datos.

12.5. Procedimiento

1. Modifique el circuito de tal forma que el pin RB0 quede con un pulsador de entrada. Como observa en el programa, solo las salidas RB1 y RB2 han sido habilitadas. Tome el programa escrito a continuación, compilelo y quémelo en pic. Presione el pulsador conectado a RB0, ¿Que diferencia observa entre el comportamiento del led en RB2 y el de RB1?, ¿Interfiere uno con el otro?. Coloque comentarios descriptivos sobre el código que indiquen de que se trata el programa principal y de que se trata la interrupción. Si la funcion `contar()` no se llama dentro del main, ¿Cuando se ejecuta este fragmento de código?.

```
#include <pic.h>
__CONFIG(0x3d71);

void retardo(int in){
    int i;
    TMR2 = 0;
    for(i=1; i<in; i++) while(TMR2 < 250);
}

void main() {
    di();//Deshabilitar interrupciones
    TRISB=0B11111001;// Pin 2 y 3 puerto B como salida
    PR2=250;    // Valor maximo de conteo para TMR2
    T2CON=0x07; // prender TMR2 y configurar preescalador
```

```
      //Configurar el bit INTEDG del Registro OPTION_REG
    INTEDG=1; //Interrupcion en RB0 en flanco de subida
    INTCON=0B00010000; //Habilita interrupciones externas en RB0

    ei();//Habilitar interrupciones

    RB1=0;

    while(1){
RB2=1;
retardo(200);
RB2=0;
retardo(200);
    }
}

void interrupt contar(void){
    INTF=0;//Limpiar la bandera de interrupcion
    RB1 = !RB1; //Cambiar el estado
}
```

2. Las funciones **ei()** y **di()** son utilizadas para habilitar y desabilitar la atención a interrupciones. ¿Por qué considera usted que se deshabilitaron al principio y se habilitaron de nuevo a la mitad del programa?. Así como estás existen otras funciones que se encuentran en librerías, explore el contenido de *pic.h* y observe si existen más funciones que le resulten útiles.

3. Probablemente haya notado que el LED en RB1 cambia muy rápidamente de estado, con solo pulsar una vez, ¿Qué explicación le puede dar?, ¿Se puede evitar?, ¿Comó?.

4. Tome el programa escrito a continuación, compilelo y quémelo en pic. En este programa se atienden dos interrupciones: la interrupción del timer y la interrupcion externa, ¿Como se diferencian en la llamada de la función de interrupción?, ¿Que espera que haga el programa?

```
#include <pic.h>
```

```
__CONFIG(0x3d71);

int contador;

void retardo(int in){
    int i;
    TMR2 = 0;
    for(i=1; i<in; i++) while(TMR2 < 250);
}

void main() {
    di();//Deshabilitar interrupciones
    TRISB=0B11111011;// Pin 2 puerto B como salida
    PR2=250;    // Valor maximo de conteo para TMR2
    T2CON=0x07; // prender TMR2 y configurar preescalador

    //Configurar fuente de clk interno para TMR0
    T0CS=0;
    //Valores de la preescala de TMR0
    PSA=0;
    PS2=1;    PS1=1;    PS0=1;

    //Habilitar la interrupcion del timer
    //y limpiar bandera
    T0IE=1;
    T0IF=0;

    //Configurar el bit INTEDG del Registro OPTION_REG
    INTEDG=1; //Interrupcion en RB0 en flanco de subida
    //Habilitar interrupcion en RB0 del registro INTCON
    //INTCON=0B00010000;
    INTE=1;
    INTF=0;

    ei();//Habilitar interrupciones

    RB2=0;
    contador=0;
```

```
    while(1){
if(contador>100){
    contador=0;
    RB2=0;
}
    }
}

void interrupt contar(void){
    di();
    if(INTF){//Presiono boton
INTF=0;//Limpiar la bandera de interrupcion
T0IF=0;
//Encender el LED
RB2=1;
//Empezar el timer
TMR0=0x00;
    }
    if(T0IF){//Se acabo el tiempo
T0IF=0;//Limpiar bandera
INTF=0;//Limpiar interr ext
contador++;
    }
    ei();
}
```

5. ¿Como lo usaría para evitar la intermitencia que se observa en el punto 3?.

6. Si en la función `contar` se comentan las lineas que colocan a cero T0IF e INTF, ¿Qué comportamiento espera en la primera, segunda y tercera interrupción recibida por el pic?

7. Si se comentan las funciones `di()` y `ei()` en la función `contar`, ¿que puede suceder?

Práctica 13

Histogramas de medición del periodo de un péndulo

13.1. Objetivos

- Usar el puerto paralelo del computador para medir en forma repetitiva el periodo de un péndulo y construir el respectivo histograma

13.2. Materiales

- Los mismos materiales utilizados en la práctica 6.
- 1 interruptor óptico.
- 1 conector "centronics" para puerto paralelo.
- 1 cable de impresora.
- montaje del péndulo simple.

13.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Manejo de timers y moduladores de ancho de pulso (PWM).

Estos temas corresponden a haber realizado las prácticas 6 , 7 y 12 y haber leido la sección 7.0 y 8.0 del manual del PIC ([12]).

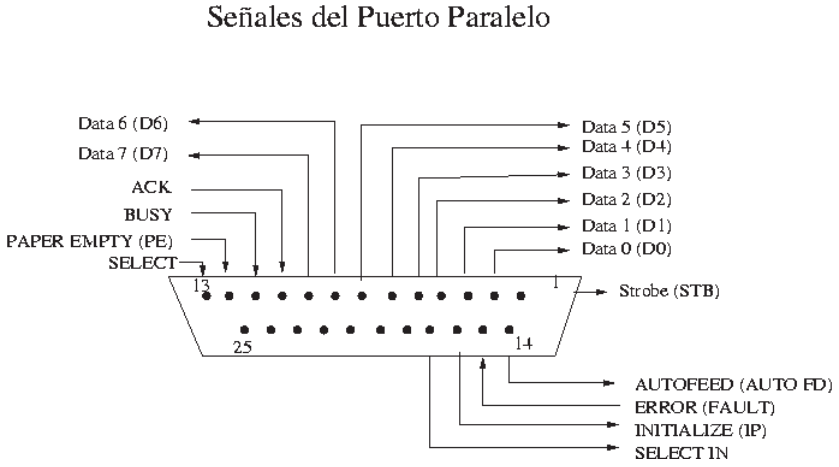


Figura 13.1: Señales del puerto paralelo.

13.4. Introducción

El puerto paralelo del computador permite el acceso a 3 registros diferentes del computador (en nuestros computadores corresponden a las direcciones 0x378, 0x379 y 0x37A). Cada uno de estos registros tienen 8 bits y pueden ser de entrada o salida de acuerdo a la Tabla 13.1.

Los registros 0x378 y 0x37A son solo de salida, mientras que el registro 0x379 tiene 5 lineas de entrada. La mayoría de las líneas de estos registros se encuentran disponibles en el conector DB25 de la impresora (ver Figura 13.1). En total se tienen 12 lineas de salida en el conector DB25 y solo 5 de entrada. Estas 5 lineas de entrada son las que se usarán para enviar las mediciones del semiperiodo del péndulo desde el pic hasta el computador.

Para leer el puerto del computador se suministrará una rutina en C que el estudiante debe modificar según lo requiera. Para procesar los datos y colocarlos en forma de histograma se puede usar el programa xmgrace.

13.4.1. Control del puerto paralelo en Linux

Linux permitirá acceder a cualquier puerto usando la llamada al sistema ioperm. Aquí están algunas partes del código en Linux para escribir 255 al puerto de la impresora:

```
#include <stdio.h>
```

LPT1	LPT2	LPT3	Bits en byte	Pines en DB25	Entrada/Salida
0x3BC	0x378	0x278	0	2	salida
			1	3	salida
			2	4	salida
			3	5	salida
			4	6	salida
			5	7	salida
			6	8	salida
			7	9	salida
0x3BD	0x379	0x279	0	NA	-
			1	NA	-
			2	NA	-
			3	15	entrada
			4	13	entrada
			5	12	entrada
			6	10	entrada
			7*	11	entrada
0x3BE	0x37A	0x27A	0*	1	salida
			1*	14	salida
			2	16	salida
			3*	17	salida
			4	NA	-
			5	NA	-
			6	NA	-
			7	NA	-

* El bit está invertido.

Tabla 13.1: Direcciones bits y pines del puerto paralelo.

```
#include <stdlib.h>
#include <unistd.h>
#include <asm/io.h>

#define base 0x378  /* printer port base address */
#define value 255   /* numeric value to send to printer port */

main(int argc, char **argv)
{
if (ioperm(base,1,1))
 fprintf(stderr,"Couldn't get the port at %x\n",base),exit(1);

 outb(value, base);
}
```

Guarde el código fuente en un archivo llamado *lpt_test.c* y compilelo con el comando:

```
# gcc -O lpt\_test.c -o lpt\_test
```

Para que el programa corra, el usuario tiene que tener privilegios para acceder a los puertos, así que usted tiene que ser root para poder correr los programas sin problemas de permisos. Si quiere hacer un programa que pueda ser corrido por cualquiera tiene que asegurarse que root sea el dueño del programa (por ejemplo realice la compilación cuando sea root), de permiso a los usuarios para ejecutar el programa y luego coloque el programa a ser ejecutado con los permisos del dueño (root) a través del comando:

```
# chmod +s lpt\_test
```

Notas sobre el código fuente: Algunas personas han reportado que por alguna razón este código no funciona en sus sistemas. Si su código tiene problemas en funcionar, trate los siguientes cambios en el código: reemplace las líneas:

`#include <unistd.h>`; y `#include <asm/io.h>`

con la línea

`#include <sys/io.h>`

y reemplace la línea

`#define base 0x378`

con

`#define base 0x0378`

Para capturar datos de 8 bits (1 byte) con las 5 líneas de entrada del puerto paralelo se puede usar el programa que se muestra a continuación

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>
#define base 0x378 /* I/O address to read */

main(int argc, char **argv){
  char test;
  unsigned char value0,value;
  float i;
  if (ioperm(base,3,1))
    fprintf(stderr, "Couldn't get the port at %x\n", base), exit(1);

  while(1){
  outb(1,base); //El pc le dice al pic: "Estoy listo !"

  do{test=inb(base+1);} //espera la respuesta del PIC
  while(test<0);
  value=inb(base+1);//Importa la primera mitad del valor
  do{test=inb(base+1);}//espera la respuesta del PIC
  while(test>0);
  value0=inb(base+1);
  value*=2; //Corra un bit a la izquierda
  value0-=128;//Quitar signo
  value0/=8;//Correr tres bits a la derecha

  i=(value+value0)*5.0/255;
  printf("Lectura de Voltaje de señal 1= %4.2f V \n",i);

  outb(0,base);

  }
}
```

Este programa supone que la señal que el pic le envía al PC se hace a través del pin 2 del conector DB25 y que los datos entran al PC por los pines 10, 12, 13 y 15 y la señal de control entra al PC por el pin 11. De acuerdo a lo anterior, la variable test se hace negativa siempre que el pin 11 sea cero y la misma variable se hace positiva siempre que el pin 11 sea uno. El pic debe programarse tal que al enviar la primera mitad del dato (4 bits) al PC ponga un cero en el pin 11 del conector DB25 y un uno en el mismo pin cuando se envie la segunda mitad del dato.

13.5. Procedimiento

Sabemos de los libros de texto que si se repite un experimento siempre bajo las mismas condiciones todos los resultados no van a ser exactamente iguales sino que van a diferir siguiendo una cierta distribución de probabilidad. Si consideramos el experimento elemental de medir el periodo de un péndulo frecuentemente se afirma en los libros que dicha distribución es gaussiana. La comprobación experimental de este hecho es difícil dado que hay que tomar un número muy grande de medidas y luego ordenarlas en un gráfica que muestra la frecuencia de ocurrencia de cada una de las medidas (histograma).

Es posible, sin embargo, diseñar un sistema que permita ir midiendo el periodo del péndulo cada vez que este realiza una oscilación y al mismo tiempo llevando dicha medición a un computador donde se vaya construyendo el histograma correspondiente.

El sistema consistirá de un pic 16f877, un interruptor óptico con su electrónica asociada y un cable de conexión al puerto paralelo del computador. La medición del tiempo se hará por medio del temporizador TMR1 del pic. El temporizador TMR1 puede mantenerse contando en forma indefinida y cada vez que pasa de su valor máximo FFFF a ceros activar una bandera de interrupción.

La medición del periodo se hace capturando el valor del TMR1 cada vez que el péndulo pasa por ejemplo a través del centro. La diferencia entre 2 capturas será igual a un semiperiodo del péndulo. Si TMR1 se ha rebasado (es decir ha llegado a FFFF y pasado a cero) es necesario sumarle FFFF a la segunda captura antes de restarle la primera captura. El pic16f877 tiene un sistema especial para capturar el valor de TMR1 que se controla a través del registro CCP1CON. Se debe configurar este registro de tal forma que cada

vez que un interruptor óptico colocado en el centro del camino del péndulo se abra se capture el valor actual de TMR1.

Una vez determinado el periodo, este dato debe enviarse a un computador a través del puerto paralelo. Dado que TMR1 tiene 16 bits la medición del periodo será un número de 16 bits. Dependiendo de la resolución que necesiten se pueden utilizar todos los bits (máxima resolución) o solo algunos de los mas significativos. En adelante se supondrá que se utilizan 8 bits.

El procedimiento sugerido es enviar las mediciones de periodo en paquetes de 4 bits desde el pic y una señal adicional de control que le indique al computador que existe un dato válido en los pines de entrada del conector DB25. El proceso de envío de los datos desde el pic empieza justo después de que el semiperiodo ha sido determinado y almacenado en algún registro del pic. El dato es mandado al puerto B y la señal de control al puerto A. Se mantiene este estado hasta que el computador lea los 4 bits más significativos de B, se desactiva la señal de control en A, se desplaza el registro B cuatro bits a la izquierda y se repite el proceso anterior para leer los bits menos significativos de la medición del semiperiodo. En el computador debe estar ejecutandose un programa que lea continuamente las lineas del puerto 0x379 y tome las acciones correspondientes de acuerdo a la señal de control.

Práctica 14

Comunicación Serial por puerto RS-232 – USART

14.1. Objetivos

- Comprender el funcionamiento y configuración del módulo USART del pic
- Familiarizar al estudiante con el uso de librerías para enviar datos desde el PIC hasta el PC y viceversa, a través del puerto RS232.

14.2. Materiales

- Los mismos materiales utilizados en la práctica 6.

14.3. ¿Qué debe saber antes del laboratorio?

Haber leido los siguientes temas:

- Generalidades de la transmisión de datos serial.
- Los niveles de voltaje en el protocolo RS-232.
- Configuración de la transmisión serial.
- Manejo de la USART.

Estos temas corresponden a haber leido la hoja de datos del componente MAX232 [15].

Consulte el Apéndice F como guía para la instalación y modificación de los paquetes necesarios.

Leer la Sección 14 de este libro y la Sección 10.0 de la hoja de datos del pic [12].

14.4. Introducción

14.4.1. Puerto RS-232

Los resultados de procesos realizados en el pic pueden ser visualizados en la pantalla del computador, es necesario establecer una comunicación. Para ello se utiliza el puerto serial del computador (RS232) que hasta ahora se ha usado solo para quemar los programas en el pic.

En el PC, el puerto RS232 tiene un conector de nueve pines (DB9) de los cuales solo usamos 3 de ellos:

- TX: transmisión de datos (pin 3)
- RX: recepción de datos (pin 2)
- COM: tierra común a ambos (puede ser el pin 5)

El computador envía datos al pic a través del pin TX y recibe datos del pic a través del pin RX. Note que hay un solo cable para transmisión de datos y un solo cable para recepción de datos, por lo tanto, el intercambio de datos entre pc y pic es de tipo **serial asincrónico**. En el pic existen al igual TX y RX, tenga en cuenta que TX_{PC} se conecta con RX_{pic} y viceversa (de lo contrario, estarían conectados dos módulos que transmiten en la misma linea, causando corto circuito y posiblemente daños a los dispositivos).

En estado inactivo el TX se mantiene en uno lógico. Cuando se quieren transmitir datos ese uno cambia a cero enviando lo que se llama el **bit start**. Se empieza a enviar una secuencia de 8 bits (el **dato** enviado). Cuando se alcanza el bit 8, TX regresa nuevamente a uno (**bit stop**) hasta que se decida enviar otro byte de información[1].

[1]El número de bits transmitido y el número de bits de stop depende de la configuración del dispositivo. Esta configuración debe ser la misma para PC y para el pic.

Para la recepción, observando a RX, se comienza por detectar el bit start, se lee la secuencia de bits que llegan hasta la llegada del bit de stop.

Cada bit enviado o recibido, incluyendo los bits de Start y Stop, tienen una duración determinada, lo cual hace que la transmisión de un byte de 8 bits tenga un tiempo de duración fijo. Esto a su vez significa que existe un número máximo de bits que se pueden transmitir por segundo, lo cual determina la **velocidad de transmisión**. Para que la comunicación se pueda realizar, ambos dispositivos deben estar configurados a la misma velocidad que para el caso de estas prácticas será 9600 baudios[2].

La configuración para la comunicación que será probada será entonces:

- Modo: Serial asíncrona
- Velocidad: 9600 baudios
- Bits de stop: 1
- Bits transmitidos: 8

A continuación se realizará la prueba de comunicación bidireccional:

- PC⇆pic

Para la práctica se usará la librería *softwarepics.tar*[3]. Para instalar esta librería copie el archivo *softwarepics.tar* en el computador y desde allí escriba el siguiente comando

```
$ tar -xvf softwarepics.tar
```

Entre otros, se creará el directorio:

- *rs232-vX/* donde están los programas que se corren desde el pc y el pic

Ubíquese en el directorio *rs232-vX*.

[2]La unidad baudios es número de símbolos transmitidos por segundo: Debido a que para este caso de comunicación, nuestra unidad de información es el bit, entonces un símbolo es igual a un bit.

[3]Ver Apéndice G para conocer su contenido.

PC⇆pic

Abra el archivo *tx1pic.c*.

```
#include <pic.h>
#include "rs232.h"
__CONFIG(0x3d71);
void main(){
  //char dato;
  char x[5];
  TRISB =0B00000000;       // Todos los bits como salida
  PORTB=0x00;
  config_UART();
  while(1){
    receiver(x,1);
    sender(x,1);
    PORTB = x[0];
  }
}
```

En este archivo:

- se inicializa el puerto B como salidas
- se inicializa la UART para el envío de datos en forma serial[4]
- se recibes serialmente un dato y se guarda en la variable X
- se envia serialmente el dato recibido
- se coloca el dato recibido en el puerto B.

Para compilar, en una consola muevase al directorio mencionado y ejecute:
`$ make -B -f Makefilepic PROGRAMA=tx1pic.c`

Esto creará el archivo *tx1pic.hex* que será programado en la memoria del pic. Programe el pic con este archivo.

Para el PC, abra el archivo *tx1pc.c*.

[4]Ver el capítulo 14 referente al funcionamiento de la USART en caso de requerir más detalles.

```
#include <stdio.h>   /* Standard input/output definitions */
#include <string.h>  /* String function definitions */
#include <unistd.h>  /* UNIX standard function definitions */
#include <fcntl.h>   /* File control definitions */
#include <errno.h>   /* Error number definitions */
#include <termios.h> /* POSIX terminal control definitions */
#include <sys/types.h>
#include <sys/stat.h>

#include "capser.h"

int main(int argc, char **argv) {
  int fd;//file descriptor (el puerto)
  char enviar[255];//Maximo de datos a recibir en un solo paquete
  char recibir[255];//Maximo de datos a recibir en un solo paquete
  unsigned char rb[255];
  int irb[255];
  int i,j,n,c;
// Apertura del puerto
  fd = abrirpuerto_ttyS0();
// Limpieza del buffer e inicializacion del puerto
  tcflush(fd,TCIOFLUSH);
  initport(fd);
  tcflush(fd,TCIOFLUSH);
  //inicializa vector enviar[] en ceros
  for(i=0;i<sizeof(enviar);i++)  enviar[i]=0;
  c = 0;
  while(c != 32){
//termina el programa con barra espaciadora + return
    for(i=8;i<16;i++){
// produce retardos en microsegundos, máximo un segundo
      usleep(900000);
      for(j=0;j<sizeof(recibir);j++) recibir[j]=0;
//Este numero debe estar entre 0 y  255 o 0x00 y 0xff
      enviar[0]=i;
//envia un byte (enviar[0]) al puerto serial
      n = write(fd, enviar, 1);
      printf("Dato enviado  = %d \n",i);
```

```
      n = read(fd, recibir, 1);
//conversion del dato recibido de char a unsigned char
      rb[0] = recibir[0];
// conversion del dato recibido de unsigned char a entero.
      irb[0] = rb[0];
//Esto siempre hay que hacerlo cuando se recibe un dato
      printf("Dato recibido = %d\n\n",irb[0]);
    }
    printf("Oprima enter para continuar//
            o barra espaciadora y enter para salir \n");
    c = getchar(); // leer tecla oprimida y guardarla en c
  }
// Limpia los buffers y cierra el puerto
  tcflush(fd,TCIOFLUSH);
  close(fd);
  return 0;
}
```

En este archivo:

- se abre el puerto /dev/ttyS0
- se borra todo lo que haya llegado o fuera a ser enviado por el puerto
- se envia un número
- se espera un tiempo
- se lee lo recibido y se repite el ciclo de envio

Compile el archivo con el comando

```
$ make -B -f Makefilepc PROGRAMA=tx1pc.c
```

Para probar la comunicación realice los siguientes pasos:

1. Presione el reset del pic
2. Con la consola en el directorio de archivos para el PC, ejecute

   ```
   $ tx1pc
   ```

Para este momento Ud. debe estar observando en el PC que el mismo dato enviado, se recibe, que es también el mismo dato que el pic despliega en el puerto B.

14.4.2. Manejo de la USART para comunicación serial

La USART (Transmisor-Receptor Sincrono-Asincrono Universal Direccionable) es un módulo del pic que permite la transmisión/recepción de datos en forma serial.

Un módulo es una construcción dentro del pic que no depende directamente del sistema principal (La ALU y la memoria). Por el contrario, funciona independiente y tiene sus propios estados y configuración, y la comunicación con el sistema principal se realiza mediante señales especiales y un bus de datos común a todos los sistemas[5]. Esto permite que el módulo USART realice el proceso de envío y recepción sin tomar tiempo o recursos adicionales del resto del pic, lo cual es útil para lograr un funcionamiento mejor de nuestro sistema respecto a rendimiento.

Las siguientes son las características, configuración y uso.

14.4.3. Características

La USART está compuesta por dos bloques funcionales separados: transmisión y recepción.

Bloque de transmisión

El transmisor está compuesto por el buffer TXREG que es el primer lugar donde se almacena el dato a enviar. Al iniciar el envío el bloque de transmisión carga el dato en el registro TSR. Este nombre es la abreviación de Registro de Desplazamiento de Transmisión y de acuerdo a su nombre lo que hace es desplazar los datos uno por uno hacia el pin de salida. La velocidad a la cual transmite (desplaza) este registro está determinada por el Generador de la Tasa de Baudios, que se encuentra colocado en el registro SPBRG, que es habilitado por la señal TXEN.

En algunos casos se requiere añadir un noveno bit a transmitir. Este noveno bit es almacenado en la señal TX9D y habilitado para su transmisión por la señal TX9. Al igual, para que los bits puedan salir por el pin RC6 del pic, deben estar configuradas la señales TRMT Y SPEN. Al terminar se habilitará la bandera de interrupción TXIF.

[5]Entiendase bus de datos como las conexiones eléctricas comunes a todos los componentes del pic a través de las cuales se transportan datos. ¡Esto significa que están conectadas también con la memoria de datos!. ¿Recuerda que existe existe una división en la memoria? Ver Sección 6.

En la Figura 14.1[6], se observa la arquitectura anteriormente descrita.

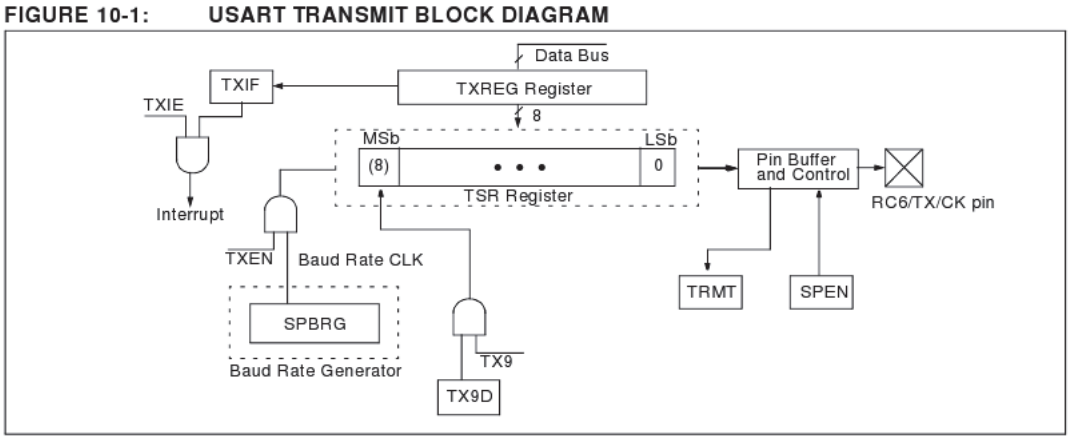


Figura 14.1: Bloque de transmisión.

Bloque de recepción

El bloque de recepción está constituido el pin RC7 y la señal de control SPEN a través de la cual llegan los datos serialmente. Se toman muestras del dato recibido entre 16 y 64 veces por cada bit,[7]con el fin de minimizar errores en la recepción. Cada dato recibido es llevado al registro de desplazamiento de recepción RSR, así hasta completar el número de bits a recibir, que pueden ser 8 o 9. El dato que terminó de llegar es almacenado en el registro RCREG y el bit adicional en RX9D, si es habilitado por RX9.

Existe una importante diferencia entre el módulo de transmisión y el módulo de recepción. En la recepción, TXREG podía almacenar un dato para enviar, sin embargo, RCREG puede almacenar 2. Esto es así debido a que en general, no se sabe en que momento van a ser recibidos datos, por lo cual se necesita espacio extra para no perder información.

Dependiendo de si se han producido errores, se habilitarán las banderas OERR y FERR, de lo contrario, se habilitará la bandera de interrupción RCIF.

[6]Imágen tomada de la hoja de datos [12], figura 10-1.

[7]Esto significa entre 16 y 64 veces más rápido que la velocidad con la cual se transmite.

En la Figura 14.2[8], se observa la arquitectura anteriormente descrita.

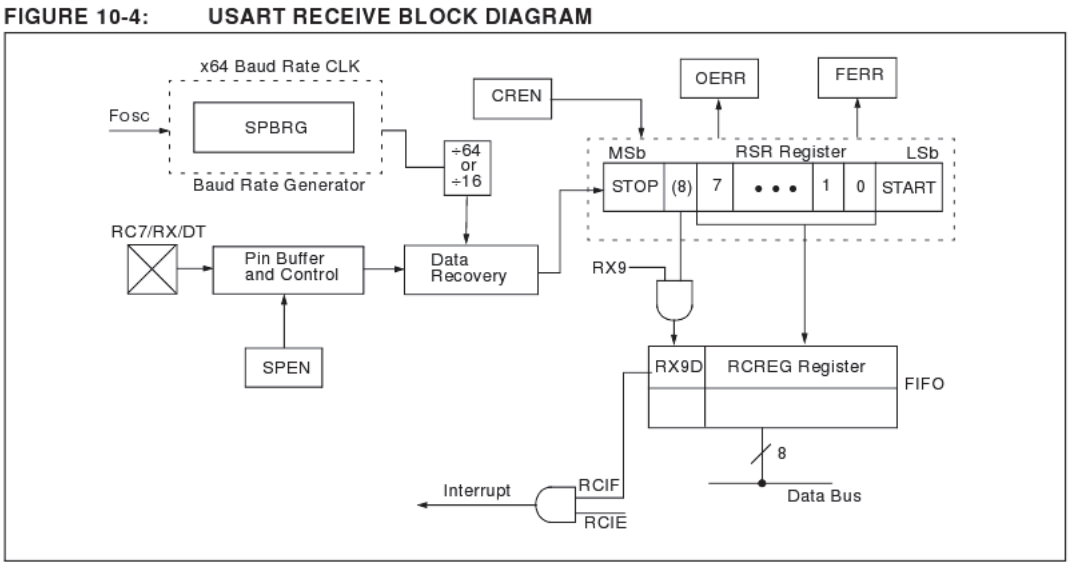


Figura 14.2: Bloque de recepción.

14.4.4. Configuración

Como se observó en la Sección 14 todo transmisión serial tiene un **modo de transmisión**, una **velocidad de transmisión** con determinadas características para la **trama**.

Modo de transmisión

La USART solo transmite en forma serial. Esta transmisión puede ser **síncrona** o **asíncrona**.

- Síncrona: Para la transmisión/recepción de datos, existen 2 señales (DT, CK) y una tierra comun (COM). Se llama síncrona porque existe una señal de clock común a ambos dispositivos. Cada vez que se realiza una transmisión de un dato, el dispositivo **master** comienza a generar la señal de clock en CK (una onda cuadrada con frecuencia configurada

[8]Imágen tomada de la hoja de datos [12], figura 10-4.

en el módulo) y en el pin DT coloca/lee el dato (bit). Este bit va cambiando con cada clock hasta transmitir/recibir todo el byte. Después de esto, queda inactivo. Tenga presente que **solo** el dispositivo master controla el clock, es decir, si el master es el PC, el clock lo controla el PC y el **slave** es el pic; si el master es el pic, el pic controla el clock y el slave es el PC; master y slave son conceptos **independientes** de transmisión y recepción. El receptor solamente determina el momento en el cual se inicia la señal de clock y lee el dato en cada cambio. Debido a que los dispositivos no pueden transmitir y recibir a la vez, este modo se caracteríza por alternar la recepción y la transmisión (Esto es denominado **Half Duplex**, doble dirección pero en un sentido a la véz). Su utilización no es muy común.

- Asíncrona: Para la transmisión/recepción de datos, existen 2 señales (TX, RX) y una tierra común (COM). Se llama asíncrona porque **no** existe señal de clock común a ambos dispositivo. Precisamente es el tipo de comunicación implementada en la Sección 14. A diferencia del modo síncrono, note que esta vez los datos se llevan a través de dos cables y no de uno. Esto permite que los módulos puedan transmitir y recibir al mismo tiempo. Es necesario que el Transmisor de uno esté conectado con el Receptor del otro para lograr la comunicación, de lo contrario se corre el riesgo de generar cortos circuitos. En este caso la transmisión se lleva a cabo, iniciandose con un nivel bajo (0 V) en el transmisor, luego envía cada uno de los bits permaneciendo en cada valor el tiempo configurado para la velocidad de transmisión. Es necesario que ambos dispositivos estén configurados igual, de otra forma los datos no se recibirán correctamente. Note además que esta clase de transmisiciones asincrónicas se basan en **una señal de inicio y final**, **una velocidad de transmisión** que determina en que momento se leerá el siguiente bit. Su utilización es mucho más amplia que el modo sincrónico debido a que la transimisión de señales de clock sobre líneas muy largas lleva problemas, además tiene la ventaja de poder transmitir y recibir al mismo tiempo (esto es denominado **Full Duplex**, doble dirección y ambos sentidos a la véz).

En adelante nos concentraremos en el modo asíncrono debido a que es más útil a permitir la comunicación de datos más eficientemente y es la más comunmente soportada por diversos dispositivos.

Velocidad de transmisión

La velocidad de transmisión especifíca que tan rápido se rotarán los bits por el puerto de salida, tal como se mencionó en el desarrollo de la Sección 14, sin embargo, solo se mencionó 9600 baudios[9]. En realidad el pic puede transmitir a varias velocidades dependiendo de la configuración interna y el cristal que se coloque. Además, existen dos modos de velocidad elegidos por el bit BRGH:

- BRGH = 1 : Baudios = $\frac{\mathrm{F_{osc}}}{16(X+1)}$.
- BRGH = 0 : Baudios = $\frac{\mathrm{F_{osc}}}{64(X+1)}$.

Donde X es el valor escogido de la columna SPBRG de la Tabla 14.1. En general se recomienda utilizar BRGH = 1, debido a que implica velocidades más altas y tasas de error menores. La Tabla 14.1[10] es un resumen de diferentes velocidades de transmisión y su porcentaje de error para el modo asíncrono de transmisión para diferentes cristales.

KBaud	$\mathrm{F_{osc}}$=20 MHz		$\mathrm{F_{osc}}$=16 MHz		$\mathrm{F_{osc}}$=10 MHz		$\mathrm{F_{osc}}$=4 MHz		$\mathrm{F_{osc}}$=3.6864 MHz	
	% error	SPBRG	% error	SPBRG	% error	SPBRG	% error	SPBRG	% error	SPBRG
0.3	-	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	0.17	207	0	191
2.4	-	-	-	-	1.71	255	0.17	103	0	95
9.6	0.16	129	0.16	103	0.16	64	**0.16**	**25**	0	23
19.2	0.16	64	0.16	51	1.72	31	0.16	12	0	11
28.8	0.94	42	2.13	33	1.36	21	3.55	8	0	7
33.6	0.55	36	0.79	29	2.10	18	6.29	6	2.04	6
57.6	3.34	20	2.13	16	1.36	10	8.51	3	0	3

Tabla 14.1: Tabla de velocidades de transmisión para la UART.

SPRBG es también un registro del pic, el cual se debe cargar con alguno de los valores anteriormente mostrados para determinar la tasa de baudios para la comunicación. Note además que en la tabla se resaltó el valor de configuración y tasa de error utilizadas para la transmisión utilizada en la Sección 14. Por supuesto, existen velocidades que disminuyen el valor del error a cero, pero corresponden también a pic que tienen un cristal más

[9]En adelante se mencionará solo la configuración para el modo asincrónico debido a que es de mayor utilidad

[10]Tomado de la tabla 10-4 de la hoja de datos [12]

lento. Por lo cual, si el objetivo del pic es solamente la comunicación de pocos datos y tareas pequeñas, se puede pensar en utilizar un cristal de menor frecuencia y asegurar la transmisión. Si por el contrario necesita transmitir una gran cantidad de datos y/o adicionalmente el pic hace muchas tareas, es recomendable buscar un cristal de mayor frecuencia y una configuración de velocidad suficientemente rápida para la aplicación con un tasa de error baja.

Trama

La trama es la secuencia que envía/recibe el módulo. Para el módo asíncrono está compuesta por:

- **Bit de start**: Es el que determina el inicio del envío. Para el puerto del pic, el estado inactivo es +5 V. Al iniciar la transmisión, se envia el bit de start que es simplemente 0 V por el tiempo programado de acuerdo a la velocidad de transmisión.

- **Dato**: El dato es transmitido/recibido bit por bit después del bit de start. Cada bit tiene la duración determinada por la velocidad del módulo. Para el pic se puede configurar la transmisión de 8 o 9 bits, este bit adicional lleva el nombre de TX9D o RX9D dependiendo del caso. Por supuesto, es necesario habilitar su transmisión mediante la configuración de los registros correspondientes. Existen varios usos que se le pueden dar a la transmisión de un noveno bit, por ejemplo:

  - Suponga que requiere realizar la transmisión de una conversión analoga/digital. Para ello ordena la conversión y lee el resultado de dos registros de ocho bits cada uno[11]. El resultado por defecto es un valor de 10 bits, lo cual implica un valor en el rango [0,1023]. Existen casos en los cuales la utilización de 9 bits es suficiente, y por lo tanto ¡podría realizar el envío/recepción de un solo dato!. Por supuesto, en el PC es necesario cambiar también la lectura y hacer que se escriba sobre una variable de tipo `int` (32 bits), en vez de una `char` (8 bits).
  - Suponga que quiere enviar un dato de 8 bits (1 byte), pero quiere asegurarse que el dato haya llegado correctamente, es decir, que

[11] Ver la configuración para una conversión en la Sección 8

todos los bits que leyo correspondan a los bits enviados. Un método común para realizar esta corroboración es la inclusión de un bit de paridad. El byte enviado tiene un número de bits, algunos de ellos 1, otros 0. El bit de paridad es 1 si hay un número impar de unos en el dato o 0 si hay un número par de unos en el dato. De esta forma, cuando se transmita el dato, y se le agregue el bit de paridad como noveno bit, el valor enviado (dato de 8 bits + bit de paridad) siempre tendrá un número par de unos. Con esto se puede realizar una verificación rápida al llegar el dato. Este sistema no es infalible y es posible que necesite realizar otros procesos para asegurarse que el dato es correcto, además, tiene que tomar decisiones sobre que hacer cuando un dato llega mal, lo cual corresponde a un programa más elaborado.

- Suponga que tiene varios pics conectados al puerto serial del PC. ¿Como se comunica con ellos? Si no organiza la comunicación se pueden generar cortos circuitos, por lo cual, es recomendable utilizar direcciones para cada dispositivo. El noveno bit puede utilizarse para distinguir entre dirección y dato, si se envía una dirección está será recibida por el pic que le corresponde y comenzarán la comunicación. Todos los otros también recibirán la dirección, pero no responderán porque no es la dirección de ellos. Esto puede realizarse de dos formas, se utiliza el noveno bit libre y se hace una codificación donde un nivel es una dato y el otro es una dirección, ó, se configura la UART[12] para que realice la verificación automática, dando la señal mediante el bit ADDEN en el registro RCSTA (ver Tabla 14.2), donde si el noveno bit es 1 corresponde a una dirección, o a un dato si es 0.

Como estos tres anteriores, hay otros usos y depende de Ud. si requiere o no utilizar el noveno bit.

- **Bit de stop**: Al enviar los bits de datos, el puerto vuelve al nivel alto por la duración de un bit, este se llama bit de stop. El pic solo soporta generar un bit de stop, y por lo general se utiliza solo uno. Sin embargo, si los módulos o el software programado no es tan eficiente para la detección o se quiere disminuir la tasa de errores afectando solo

[12]Notese que se ha llamado ahora UART y no USART ya que no se utiliza el modo sincróno.

un poco la velocidad de transmisión, entonces se colocan dos bits de parada y se compensa este tiempo en el pic.

Recuerde que debido a que la comunicación es asíncrona, es necesario que ambos dispositivos tengan la misma configuración, esto implica en esta sección el mismo número de bits transmitidos y el mismo número de bits de stop. No se menciona el bit de parada ya que no es común modificarlo.

14.4.5. Uso

Los siguientes son los registros que necesita configurar para realizar la comunicación serial asíncrona utilizando la USART.

TXSTA: El registro TXSTA es el registro de Status y Control del módulo de transmisión. La Tabla 14.2 muestra la configuración de este registro.

TXSTA							
R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	—	TX9D
b7	b6	b5	b4	b3	b2	b1	b0

R = Bit legible
W = Bit escribible
U = Bit sin implementar, al leer se obtiene '0'
-n = Valor cuando se ejecuta el POR Reset

Tabla 14.2: TXSTA: Registro de status y control de la transmisión.

La función que cumple cada bit en el registro TXSTA se describe a continuación:

- Bit 7:**CSRC: Bit de selección de la fuente de clock para la USART**
  - Modo asíncrono: No importa.
  - Modo síncrono:
    - 1 = Modo master (clock generado internamente por el generador de baudios)
    - 0 = Modo slave (clock de fuente externa)

- Bit 6:**TX9: Habilita el envío del bit 9**
  - 1 = Selecciona el envío de 9 bits
  - 0 = Selecciona el envío de 8 bits
- Bit 5: **TXEN: Habilita la transmisión**
  - 1 = Habilita la transmisión
  - 0 = Deshabilita la transmisión

  Nota: SREN/CREN sobreescriben TXEN en modo sincrónico.
- Bit 4: **SYNC: Bit de selección de modo de la USART**
  - 1 = Modo síncrono
  - 0 = Modo asíncrono
- Bit 3: **Sin implementar: Al leer se obtiene 0**
- Bit 2: **BRGH: Bit de selección de tasas altas de baudios**
  - Modo asíncrono
    - 1 = Velocidad alta[13]
    - 0 = Velocidad baja
  - Modo síncrono: No usa este bit.
- Bit 1: **TRMT: Bit de status del registro de desplazamiento**
  - 1 = El registro está vacío
  - 0 = El registro está lleno
- Bit 0: **TX9D: Noveno bit de transmisión, puede ser un bit de paridad**

RCSTA: El registro RCSTA es el registro de Status y Control del módulo de recepción. La Tabla 14.3 muestra la configuración de este registro.

[13]Es recomendable utilizar baja velocidad, debido a que las tasas de error de transmisión son en general menores.

RCSTA							
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
b7	b6	b5	b4	b3	b2	b1	b0

R = Bit legible
W = Bit escribible
U = Bit sin implementar, al leer se obtiene '0'
-n = Valor cuando se ejecuta el POR Reset

Tabla 14.3: RCSTA: Registro de status y control de la recepción.

La función que cumple cada bit en el registro RCSTA se describe a continuación:

- Bit 7:**SPEN: Bit de habilitación del puerto serial**
  - 1 = Puerto serial habilitado (configura el RC7 y RC6 como pines del puerto serial)
  - 0 = Puerto serial deshabilitado
- Bit 6:**RX9: Habilita recepción del bit 9**
  - 1 = Selecciona la recepción de 9 bits item 0 = Selecciona la recepción de 8 bits
- Bit 5:**SREN: Habilita la recepción de un solo dato**
  - Modo asíncrono: No importa
  - Modo síncrono - master
    - 1 = Habilita la recepción de un solo dato
    - 0 = Deshabilita la recepción de un solo dato

    Este bit se pone en 0 después de completada la recepción
  - Modo síncrono - slave: No importa
- Bit 4:**CREN: Habilita la recepción en modo contínuo**
  - Modo asíncrono
    - 1 = Habilita la recepción contínua
    - 0 = Deshabilita la recepción contínua
  - Modo síncrono

    - 1 = Habilita la recepción contínua hasta que el bit CREN es puesto en 0 (CREN sobreescribe SREN)
    - 0 = Deshabilita la recepción contínua

- Bit 3:**ADDEN: Habilita la detección de dirección**
    - Modo asíncrono bit 9 (RX9 = 1)
        - 1 = Habilita la detección de dirección, habilita la interrupción y carga del buffer de recepción cuando el bit8 de RSR es puesto en 1
        - 0 = Deshabilita la detección de dirección, todos los bytes son recibidos, el noveno bit puede ser usado como un bit de paridad
- Bit 2:**FERR: Bit de error de trama**
    - 1 = Error en la trama (puede ser actualizado leyendo RCREG y recibir el siguiente byte válido)
    - 0 = No revisa error de trama
- Bit 1:**OERR: Bit de error de sobrecarga**
    - 1 = Error de sobrecarga (puede ser limpiado poniendo en 0 el bit CREN
    - 0 = No revisa error de sobrecarga
- Bit 0:**TX9D: Noveno bit de recepción, puede ser un bit de paridad, pero debe ser calculado por el firmware escrito por el usuario**

Como puede observar, el control de la recepción requiere más trabajo de control que la transmisión, esto se debe a que el pic transmite en cualquier momento y el PC tiene recursos amplios para recibir cientos de datos y mantenerlos en el buffer de entrada, sin embargo, en general no se sabe en que momento va a iniciar una transmisión el PC, ni cuantos datos se van a transmitir. Esta es precisamente la razón por la cual el buffer de recepción es más grande que el buffer de transmisión en el pic.

Es por lo tanto una buena idea en la programación mantener el número de datos enviados del PC al pic al mínimo posible y es aún más fácil si se

transmite de a un solo dato a la vez. Si realizo el programa del pic de tal forma que se reciba de a un solo dato, pero sea suficientemente rápido para volver a la recepción antes del comienzo de la recepción del segundo dato y así sucesivamente, entonces no se tendrán problemas con la recepción de un número múltiple de datos. Esto implica por supuesto que el pic está dedicado a enviar datos y poco tiempo le dedica a otras tareas. Note que esta es precisamente la opción que se tomó para la transmisión de datos en la librerías *rs232-X*[14], ya que a pesar de usar la UART para generar la secuencia de envío de principio a fin, es en todo caso el software el que está listo para recibir. **¡Si por alguna razón el PC envía, y el pic está ejecutando otra parte del software que no corresponda a la recepción, entonces esos datos se perderán!**

El bit CREN de RCSTA habilita la recepción en modo contínuo, esto significa que continuará leyendo el puerto constantemente y por lo tanto la probabilidad de pérdida de datos como en el caso anterior es mínima. Sin embargo, esto implica una lógica de control diferente, ya que el software se encontrará en la ejecución de cualquier instrucción, es decir, no necesariamente está pendiente de una recepción serial, entonces el mejor método es habilitar la interrupción RCIE del módulo de recepción para guardar el dato en el momento que sea recibido[15]. Cuando se detecte la recepción del primer dato esté será almacenado en el registro de desplazamiento para la recepción. Una vez completado se pasará al registro RCREG[16]. Como está en funcionamiento contínuo entonces el módulo continúa trabajando y si detecta la recepción de un segundo byte, este llega al registro de desplazamiento. Cuando se completa, se pasa al segundo nivel de RCREG.

Ahora supongamos que se detecta la llegada de un tercer byte y no se ha leido aún ningun dato de RCREG, este se llevará al registro de desplazamiento, pero cuando se termine de recibir, no habrá donde almacenarlo.

[14]Ver Sección 14

[15]Esto significa que tendrá que agregar las instrucciones correspondientes a la atención de la interrupción de recepción en el programa del pic, de acuerdo a lo mostrado en la Sección 12. En esta rutina ¿donde almacenaría el dato que llega para no alterar el funcionamiento del programa principal? Una sugerencia podría ser colocar un arreglo `char` en el programa principal, tener una variable contadora (apuntador) que determine cual es la última posición vacía del arreglo y escribir allí

[16]Recuerde que RCREG es un registro de dos bytes, y además su lectura es **FIFO** (First-In First-Out), el primero que llega es el primero que sale, por lo cual si hay dos datos en RCREG, al leerlo la primera vez obtendremos el primer valor almacenado y al leerlo una segunda vez obtendremos el segundo.

Esta es precisamente la condición que genera el bit de error OERR o error de sobrecarga, ya no hay más espacio para recibir.

Por otro lado, si en la trama el bit de stop que se supone debe ser un nivel alto, se detecta como un nivel bajo, entonces significa que existe un error en la configuración o en la transmisión y es cuando se activa el bit de error FERR. Las acciones a tomar al detectar un error en la recepción deben determinarse por software ya sea en el programa principal o en la atención de la interrupción relacionada con el módulo.

Existe la posibilidad que Ud. no esté interesado en verificar ninguna de las dos, que es el caso en el cual se deshabilitan estas dos verificaciones de error propias del módulo.

TXREG: Es el registro donde se coloca el dato que el módulo de trasnsmisión va a enviar. Tiene el tamaño de 1 byte de 8 bits.

RCREG: Es el registro donde el módulo de recepción coloca el último byte completo recibido. Tiene el tamaño de 2 bytes de 8 bits cada uno. Recuerde que este es un registro **FIFO**, por lo cual, la primera vez que lo lea obtendrá el primer byte de la secuencia leida, y la segunda vez que lo lea obtendrá el segundo byte que llegó.

SPBRG: Es el registro cuyo valor determina la velocidad a la cual funcionará el módulo generador de baudios, es decir, determina la velocidad de transmisión. El valor de este registro debe corresponder a los indicados en la Tabla 14.1[17]

[17]Recuerde que esta tabla es solo válida cuando BRGH = 1, es decir, transmisión a alta velocidad.

Los bits **TXIF**, **TXIE** corresponden a la bandera y la habilitación de la bandera de las interrupciones de transmisión. Los bits **RCIF**, **RCIE** corresponden a la bandera y la habilitación de la bandera de las interrupciones de recepción. Depende de su implementación la administración de su funcionamiento, sin embargo, la forma de implementación de atención a interrupciones la puede encontrar en el Capítulo 12.

Para el envío y recepción

Las siguientes es la secuencia que debe colocar para enviar un dato, recibir un dato o recibir un dato con detección de dirección serialmente en modo asíncrono.

Envío de un dato

1. Inicialice el SPBRG al valor correspondiente a la velocidad deseada. Si va a trabajar con velocidades altas, coloque BRGH = 1
2. Habilite el puerto serial asíncrono colocando SYNC = 0 y SPEN = 1
3. Si se desean interrupciones coloque TXIE = 1
4. Si se desean 9 bits, entonces coloque TX9 = 1
5. Habilite la transmisión colocando TXEN = 1, lo cual también habilitará la bandera TXIF
6. Si se va a transmitir el noveno bit, carguelo en TX9D
7. Cargue el valor a transmitir en TXREG
8. Si usa interrupciones asegurese que GIE y PEIE del registro INTCON estén habilitados

Recepción de un dato

1. Inicialice el SPBRG al valor correspondiente a la velocidad deseada. Si va a trabajar con velocidades altas, coloque BRGH = 1
2. Habilite el puerto serial asíncrono colocando SYNC = 0 y SPEN = 1
3. Si se desean interrupciones coloque RCIE = 1
4. Si se desean 9 bits, entonces coloque RX9 = 1

5. Habilite la recepción colocando CREN = 1
6. El bit RCIF se pondrá en 1 cuando se complete la recepción y se generará la interrupción si RCIE está en 1
7. Lea el registro RCSTA para determinar el noveno bit y si ocurrió algún error en la recepción
8. Lea el registro RCREG para leer el dato
9. Si ocurrió algun error, límpielo colocando el bit CREN en 0
10. Si usa interrupciones asegurese que GIE y PEIE del registro INTCON estén habilitados

Recepción de un dato con detección de dirección

1. Inicialice el SPBRG al valor correspondiente a la velocidad deseada. Si va a trabajar con velocidades altas, coloque BRGH = 1
2. Habilite el puerto serial asíncrono colocando SYNC = 0 y SPEN = 1
3. Si se desean interrupciones coloque RCIE = 1
4. Si se desean 9 bits, entonces coloque RX9 = 1
5. Coloque la detección de direccion ADDEN = 1
6. Habilite la recepción colocando CREN = 1
7. El bit RCIF se pondrá en 1 cuando se complete la recepción y se generará la interrupción si RCIE está en 1
8. Lea el registro RCSTA para determinar el noveno bit y si ocurrió algún error en la recepción
9. Lea el registro RCREG para leer el dato
10. Si ocurrió algun error, límpielo colocando el bit CREN = 0
11. Si el dispositivo fue direccionado, coloque ADDEN = 0 para que se puedan leer los siguientes datos o direcciones y atienda la interrupción de la CPU

Note que durante la configuración existen varias cosas en común para todos los procesos, es por esta razón que se creo en el Capítulo 14 la función `config_UART()`, donde se inicializa una vez el módulo y luego solo se implementan las acciones de enviar y recibir.

14.5. Procedimiento

1. Utilizando las funciones de la Sección 14.4.1, referente al envío y recepción de datos PC⇆pic, modifique:

   a) Aumente la velocidad de envío y recepción de datos, ¿Es necesario modificar ambos programas?

   b) Aumente simultáneamente el número de bits enviados y recibidos en el PC y el PIC.

   c) Disminuya simultáneamente el número de bits enviados y recibidos en el PC y el PIC.

   d) Coloque un número mayor de datos enviados por el PC que los leidos en el pic. ¿Qué sucede?, ¿Por qué?

   e) Coloque un número menor de datos enviados por el PC que los leidos en el pic. ¿Qué sucede?, ¿Por qué?

   f) Coloque un número mayor de datos enviados por el pic que los leidos en el PC. ¿Qué sucede?, ¿Por qué?

   g) Coloque un número menor de datos enviados por el pic que los leidos en el PC. ¿Qué sucede?, ¿Por qué?

2. Modifíque el programa del PC utilizado en el punto 1, de tal forma que los datos obtenidos de ADRESL y ADRESH queden en una sola variable que barrería los números 0 a 1024 a medida que el voltaje en A0 varía entre 0 V y 5 V.

   Sugerencia: defina una nueva variable entera

   ```
   temp = ADRESH*256+ADRESL
   ```

3. Escoja tres o cuatro valores igualmente espaciados de Voltaje en A0. Registre los correspondientes números de la variable `temp`. A partir de estos datos modifique el programa anterior para que `temp` muestre

valores de 0,000 a 5,000 a medida que se cambia el voltaje en A0 desde 0 V hasta 5 V.

4. Haga una curva de calibración, es decir, una gráfica del número mostrado en `temp` en el eje Y (después de la modificación) vs el voltaje en A0 en el eje X medido con un voltímetro. Use al menos 20 datos que barran todos los valores desde 0 V a 5 V.

5. Dibuje sobre la curva de calibración la curva ideal $y = x$. ¿Hay diferencias entre la curva de calibración y la curva ideal? ¿Qué tipos de errores están presentes?, ¿Sistemáticos, aleatorios o ambos?.

6. Basado en la anterior gráfica que se puede decir acerca de la precisión y de la exactitud del voltímetro que acaba de construir con el pic. ¿Es posible calcular números para estos 2 parámetros?. Si la respuesta es positiva, hágalo.

7. Teniendo en cuenta que el conversor A/D del pic es de 10 dígitos, calcule la mínima resolución de su voltímetro. Según este cálculo, ¿Cuántos dígitos de `temp` son realmente confiables?. Compare con lo obtenido en los puntos 5 y 6.

8. Aumente la velocidad de funcionamiento de la UART. Recuerde que para esto ambos dispositivos tienen que modificarse, por lo cual, es necesario cambiar el parámetro que modifica la velocidad del puerto del PC en la función `initport()`. La línea correspondiente tiene la siguiente apariencia actual

```
int initport(int fd) {
...
  cfsetospeed(&options, B9600);//cambiar está línea
...
}
```

en el archivo *./pc/ser/captain/capser.c* busque un balance entre la velocidad a utilizar y el margen de error que se puede presentar en la comunicación. Solo algunos valores específicos de velocidad son posibles.

9. Implemente el sistema de detección de errores por bit de paridad ¿Considera que es un método confiable?, ¿Que sucede si se presenta error en dos bits de la trama? La función `initport()` configura el puerto para recibir 8 bits sin paridad, por lo cual tendrá que cambiar la línea

```
int initport(int fd) {
...
  options.c_cflag &= ~PARENB;//cambiar está línea
...
```

   Así Ud. no realiza la verificación en el PC, y los datos pueden seguir leyendose como `char`.

10. Implemente el envió de 9 bits de datos procedentes de la conversión análoga-digital del conversor. Teniendo en mente alguna aplicación que requiera ¿Considera 9 bits suficiente resolución? Recuerde que las funciones de ejemplo implementadas en el PC solo leen datos de tamaño `char` porque la configuración del puerto en `initport()` es recibir 8 bits. Esto significa que tendrá que cambiar de nuevo la función, esta vez

```
int initport(int fd) {
...
  options.c_cflag |= CS8;;//cambiar está línea
...
```

    recordando que en este caso no se transmite bit de paridad.

11. Con el fin de realizar otros procesos, utilice las interrupciones para el control de la transmisión de datos. Una vez logrado, realice la recepción de datos utilizando las interrupciones.

    Sugerencia: Pruebe inicialmente con la transmisión y recepción de un dato. Luego, pruebe la recepción de un dato y la transmisión de varios, finalmente la recepción y transmisión de varios datos. Piense en una aplicación; la cantidad de datos con la cual haga las pruebas debe estar de acuerdo con las necesidades de comunicación de su aplicación.

12. Ahora conecte dos pics, de tal forma que TX y RX estén cruzados respecto al PC, pero que estén directos entre pics. Realice la implementación de la detección de dirección mediante el noveno bit. La implementación mediante el hardware especial de la UART o por software es su decisión.

    Nota: Recuerde que errores en la conexiones puede llevar a cortos circuitos, por lo cual, es recomendable que haga un diagrama preliminar antes de conectar.

14.6. Anexos

Respecto a los paquetes escritos, el proposito de estos programas en básicamente educativo, sin embargo, le puede resultar interesante utilizar alguna de las ideas aquí escritas (verbalmente o en codigo) para su proyecto. Por lo cual, es libre de utilizarlo y modificarlo a su gusto.

Si lo utiliza para un mayor proyecto, tenga especial cuidado con la posibilidad de errores en la transmisión, a pesar que las funciones declaradas el PC retornan valores dependiendo de si fue exitosa o no la escritura, en ningun punto se realiza una verificación de estos valores o sobre los datos transmitidos. Se dejó en esta forma por simplicidad para nosotros y posiblemente por utilidad para Vd. (Vd. conoce mejor los requerimientos en su sistema).

El pic se comunica usando la UART pero no utiliza interrupciones. Tenga especial cuidado en el sincronismo de los dos programas, o puede perder información o esperar respuestas que el PIC no recibe.

Parte III

Apéndices

Apéndice A

El Informe

A.1. Informe Regular

El informe regular de las prácticas de este laboratorio deberá tener al menos una introducción, el procedimiento y análisis, las conclusiones y la bibliografía . La introducción debe ser una composición coherente y bien escrita de los conceptos básicos que son necesarios para el desarrollo de la respectiva práctica y que demuestre la comprensión del tema. Para escribir la introducción se puede utilizar el material visto en la clase teórica, la introducción misma de la práctica así como también lo que se investigue por cuenta propia (altamente recomendado). Bajo ninguna circunstancia la introducción podrá ser una colección de retazos de diferentes fuentes sin ninguna conexión aparente, ni mucho menos copias fieles de cualquier fuente particular de información. Se recomienda escribir la introducción después de haber desarrollado el procedimiento y debe tener al menos una página de extensión y no más de dos páginas de extensión.

En la sección de procedimiento y análisis se deben resolver cada uno de los puntos requeridos en la guía del laboratorio y escribir los análisis y comentarios que el estudiante considere necesarios. En esta sección, también deben colocarse los diagramas y dibujos que en opinión del estudiante ayuden a comprender los análisis hechos. No hay límite de páginas para esta sección. Las conclusiones deben ser breves y tener una conexión clara con los análisis realizados en la sección previa. Finalmente, en la bibliografía debe aparecer todo el material que hayan consultado, libros, revistas, sitios de internet, manuales, etc.

A.2. Informe del Proyecto Final

El informe del proyecto final deberá tener las siguientes secciones

- TITULO: Título del proyecto
- RESUMEN: Se debe explicar en términos generales que problema se pretende resolver con el circuito, descripción en una línea ó dos del circuito y el cumplimiento de los objetivos iniciales del proyecto
- DESCRIPCIÓN DEL CIRCUITO: Esta sección debe incluir un esquema eléctrico detallado del circuito (mejor si lo pueden hacer en un programa como ORCAD, EAGLE o similar) y descripción detallada del funcionamiento del circuito. En esta sección debe incluirse una foto de alta resolución del montaje. Debe entregarse también una foto en formato digital.
- CÓDIGO FUENTE: En esta sección debe incluirse el código fuente en lenguaje C debidamente comentado. Debe explicarse el programa en detalle para lo cual puede valerse de diagramas de flujo.
- DIFICULTADES ENCONTRADAS y SUGERENCIAS: En esta sección deben explicar en detalle aquellos problemas que en su concepto fueron los mas difíciles de solucionar así como que aspectos que quedan pendientes por ser solucionados o mejorados en el proyecto si es el caso.
- BIBLIOGRAFÍA: Todo el material que hayan consultado, libros, revistas, sitios de internet, manuales, etc debe aparecer en esta sección.

NOTA IMPORTANTE: Los grupos que tengan proyectos para los cuales existía un informe escrito antes de que iniciaran a trabajar (EPR, posicionador de sonda, cargador) deben leer cuidadosamente dichos informes y agregar aquella información que en su concepto era necesaria para el desarrollo exitoso del proyecto pero que no fue incluida originalmente. En lo posible deben conservar el formato que se propone en esta guía.

Apéndice B

Recomendaciones para el Uso del Laboratorio

B.1. Fuente de voltaje PROTEK 3033B

- Las fuentes duales pueden trabajar ya sea como fuentes de voltaje o como fuentes de corriente. Para usarlas como fuentes de corriente lleve la perilla para ajustar voltajes al máximo y la perilla de corrientes al mínimo. Conecte la carga y empiece a mover la perilla de corriente. El valor de la corriente actual en la carga aparecerá en el display derecho de la fuente. Para usar la fuente dual como fuente de voltaje lleve la perilla de ajustar corrientes a un valor superior al consumo de corriente del circuito (si está seguro, puede llevarla al máximo) y gradué la perilla de ajustar voltajes de acuerdo al valor requerido.

- La fuente de voltaje puede ser conectada a tierra usando una pequeña lámina de metal ubicada en los bornes de ambos lados de la fuente. El borne positivo o el borne negativo de cada lado puede ser colocado a tierra en forma independiente.

B.1.1. Prueba de la fuente de voltaje

La fuente puede probarse conectándola directamente al multímetro en la escala de voltaje. La lectura del voltímetro y el display de la fuente deben coincidir.

Figura B.1: Fuente de voltaje PROTEK 3033B

B.2. Multimetro PROTEK 506

Si el multímetro quiere ser usado como voltímetro deberá colocarse en una de las 3 escalas diseñadas para tal fin y deberá conectarse en PARALELO con el componente a través del cual se quiere medir el voltaje. Además las puntas del multímetro deberán estar conectadas a los bornes rotulados COM y V.

Si el multímetro quiere ser usado como amperímetro deberá colocarse en una de las 3 escalas diseñadas para tal fin y las puntas deben estar conectada entre el borne COM y uno de los bornes rotulados como 20 A, mA ó μA según sea el caso. Tenga presente que en el borne mA la corriente nunca puede sobrepasar los 500 mA y en el borne μA la corriente nunca puede sobrepasar los 400 μA. Para hacer una medición de corriente es necesario "cortar" el circuito en el punto donde la corriente quiere ser medida e intercalar el multímetro entre los dos extremos resultantes, es decir el multímetro será colocado en SERIE.

Si el multímetro quiere ser usado como medidor de resistencias (ohmetro) deberá colocarse en la escala respectiva y asegurarse que el circuito ha sido DESCONECTADO de la fuente de alimentación.

Este multímetro también puede medir capacitancias, inductancias, frecuencias, temperatura y el estado de ciertos diodos. Información acerca de

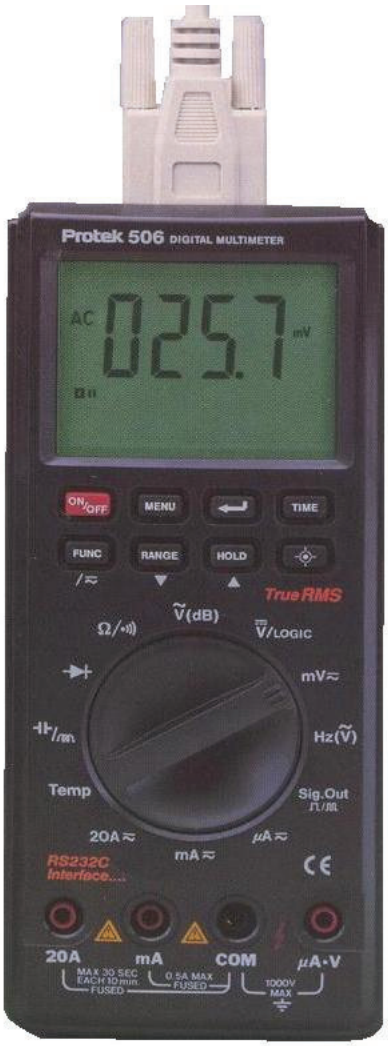


Figura B.2: Multimetro PROTEK 506

como medir estas cantidades las puede encontrar en el manual del multímetro.

B.2.1. Prueba del multimetro

Los fusibles del multímetro pueden probarse colocandolo en la función de medir resistencias. Se conecta un cable entre el borne COM y 20A y luego entre COM y mA. En ambos casos la lectura debe ser igual a cero.

B.3. Protoboard

El protoboard es una base sobre la cual se pueden colocar los diferentes componentes electrónicos de un circuito dado. El protoboard proporciona no solo una forma de asegurar los componentes sino las conexiones eléctricas necesarias entre cada uno de estos componentes. El protoboard en la figu-

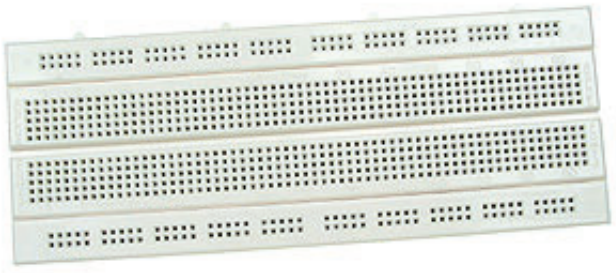


Figura B.3: Protoboard

ra B.3[1] se compone de 4 secciones separadas por cierto espacio que sirve como aislante entre cada una de ellas; las 2 secciones externas se usan generalmente para conectar las fuentes de alimentación y lineas comunes como las tierras. En las 2 secciones internas se colocan los diferentes componentes electrónicos como resistencias, integrados etc. Para comprender como están conectados cada uno de los agujeros dentro de las diferentes secciones refierase a la Figura B.4[2]. En dicha figura se ha levantado la capa superior externa

[1]Figura tomada de [19]
[2]Figura tomada de [20]

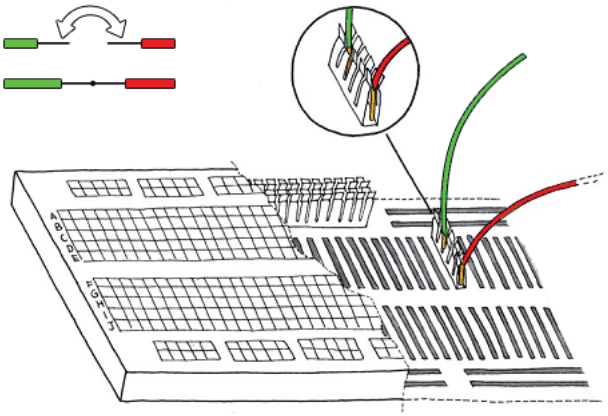


Figura B.4: Conexiones internas de un protoboard

del protoboard quedando al descubierto sus contactos metálicos. Note como los agujeros de las secciones externas están eléctricamente conectados en una dirección y los de las secciones internas en dirección perpendicular a la primera. Algunos protoboard tienen una división en las secciones externas justo en la mitad por lo que es conveniente colocar un cablecito allí que una eléctricamente ambas mitades.

Para facilitar el trabajo de localización de fallas y poder modificar fácilmente el circuito montado en un protoboard es conveniente:

- Montar el circuito en el protoboard lo mas parecido posible a su correspondiente diagrama eléctrico.
- Colocar las señales de entrada a la izquierda y las de salida a la derecha.
- Usar convención de colores.La convención de colores establece que los cables utilizados para hacer conexión a:
  - tierra debe ser negro.
  - +Vcc debe ser rojo
  - +15 V debe ser amarillo (usado para alimentar circuitos operacionales).
  - -15 V debe ser azul (usado para alimentar circuitos operacionales).

Cuando no hay +15V y -15V es común conectar las dos secciones externas del protoboard de tal forma que en ambas se disponga de +Vcc y tierra.

Los cables que se usen en el protoboard deben ser de calibre 18 y deben pelarse cerca de 1 cm en cada extremo de tal forma que toquen el fondo del protoboard cuando sean insertados dentro de los agujeros.

En lo posible tratar que cables y componentes queden a ras de la superficie del protoboard. Cables y componentes innecesariamente largos pueden conducir a cortos u oscilaciones indeseadas.

Para mejorar la estabilidad mecánica de un montaje en protoboard y para eliminar cierto tipo de ruidos es conveniente entorchar los cables de las señales de entrada y salida, cables de alimentación y cualquier otro par de cables que se conecte al protoboard.

B.4. Osciloscopio TEKTRONIX TDS 210

Figura B.5: Oscilocopio TEKTRONIX TDS 210

El TDS 210 es un osciloscopio digital de 60 MHz y 2 canales. Entre las funciones mas importantes para este laboratorio están los menús de canales (CH1 y CH2), el menú MEASURE, el menú CURSOR, y el menú DISPLAY. Entre la perillas de ajuste de mayor uso están las perillas VOLTS/DIV de ambos canales, la perilla SEC/DIV, las perillas POSITION y el botón AU-

TOSET. Este último botón es sin duda el de mayor uso pues permite ajustar automáticamente el osciloscopio de tal forma que aparezca en la pantalla la señal presente a la entrada de alguno de los canales.

Las perillas VOLT/DIV permiten cambiar la resolución vertical del osciloscopio mientras que la perilla SEC/DIV permite cambiar la escala horizontal o base de tiempo del osciloscopio. Las perillas POSITION permiten mover la señal vertical o horizontalmente a voluntad. Los botones menú muestran un menú de posibilidades en la parte derecha de la pantalla cuando son presionados. La selección de una de esas posibilidades se hace mediante los botones que se encuentran a la derecha de la pantalla.

El menú de canales (CH1 y CH2) permite escoger el tipo de acoplamiento (DC, AC o GND), si se limita o no el ancho de banda, si el cambio de resolución con el botón Volts/Div es grande o pequeño y si la sonda es directa o atenuada ya sea por 10, 100 ó 1000.

A través del menú MEASURE es posible medir voltajes medios, RMS y pico-pico. También se puede medir el periodo y la frecuencia de la señal. Antes de hacer la medición es necesario escoger el canal que quiere medirse. Esto se hace a través del primer ítem del menú MEASURE.

El menú CURSOR permite la visualización de 2 líneas horizontales (cursores de voltaje) o de 2 líneas verticales (cursores de tiempo). La posición de estas lineas puede variarse a voluntad con los botones verticales POSITION. La posición de ambos cursores es mostrada en la pantalla (en voltajes o tiempos según sea el caso) así como también su diferencia en el reglón rotulado como DELTA.

Del menú DISPLAY la opción que será utilizada con mas frecuencia en este laboratorio es la de formato. Esta opción permite escoger entre formato Y(t), es decir, la gráfica usual de voltaje vs tiempo o formato XY o gráfica de voltaje en Y (CH2) vs Voltaje X(CH1).

B.4.1. Prueba Osciloscopio

Conectar una sonda a alguno de los canales del osciloscopio. Conectar la punta de la sonda a la lámina marcada PROBE COMP en el frente del osciloscopio. Oprimir el botón AUTOSET. Una señal cuadrada de 5 V a 1 KHz deberá aparecer en la pantalla del osciloscopio.

B.5. Generador TEKTRONIX CFG 280

El generador de funciones del laboratorio puede producir ondas sinusoidales, cuadradas, triangulares, diente de sierra y otras en un rango de frecuencias que va de 0.1 Hz a 11 MHz. La frecuencia se controla mediante botones tales como el "frequency dial" que permite el ajuste de continuo de frecuencias y "freq adj fin" que permite un ajuste aun mucho mas fino. Se tiene además dos botones (multiplier) que suben o bajan la frecuencia en potencias de 10 cada vez que son presionados.

Figura B.6: Generador de funciones TEKTRONIX CFG 280.

La amplitud de la señal se controla mediante el botón "amplitud" y su nivel DC mediante el botón "DC offset". Una hilera de 5 botones en la parte superior derecha del generador permite variar la forma de onda. La salida principal del generador es el tercer conector BNC de izquierda a derecha y tiene una impedancia de salida de 50 ohmios. Es quizas el único conector que se usará para este curso. Otras funciones del generador además de las ya presentadas serán explicadas a medida que avance el curso o pueden leerse directamente en el manual.

B.5.1. Prueba del generador

Conectar una sonda directa entre el BNC MAIN OUT del generador y el canal del osciloscopio que se encuentre activo. Presionar AUTOSET. Una vez que la forma de onda aparezca en el osciloscopio cambie amplitud y frecuencia en el generador y observe el efecto de los cambios en la pantalla del osciloscopio.

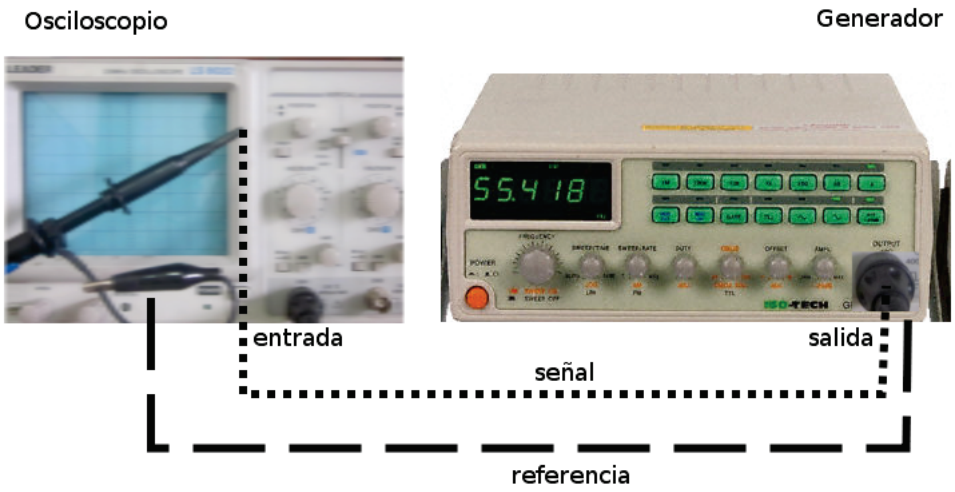


Figura B.7: Equipos con doble conexión.

B.6. Conexiones entre equipos y conexiones a tierra

La mayoría de los equipos del laboratorio quedan físicamente conectados a tierra (la cual es un muy buen conductor) cuando son enchufados en una toma de corriente. Esta conexión es necesaria para evitar posibles choques eléctricos que podrían resultar de una alta diferencia de potencial entre el equipo conectado y la tierra física.

Equipos como el osciloscopio o el generador necesitan de 2 conexiones o cables para transportar o recibir la señal (Figura B.7). De hecho, cuando se conecta una sonda entre los conectores BNC de generador y osciloscopio, uno de los cables es el conductor central del coaxial y el otro cable lo constituye la malla que rodea el cable central. Sin embargo, dado que el conductor externo de los conectores BNC en ambos equipos están conectados a tierra bastaría un alambre entre los centros de los BNC para llevar señal del generador al osciloscopio (Figura B.8). Es recomendable comprobar este hecho en el laboratorio.

Cuando se hace una conexión directa entre generador y osciloscopio usualmente se hacen ambas conexiones a través de una sonda que solo puede conectarse de una forma, es decir no es posible invertir el cable de tierra con

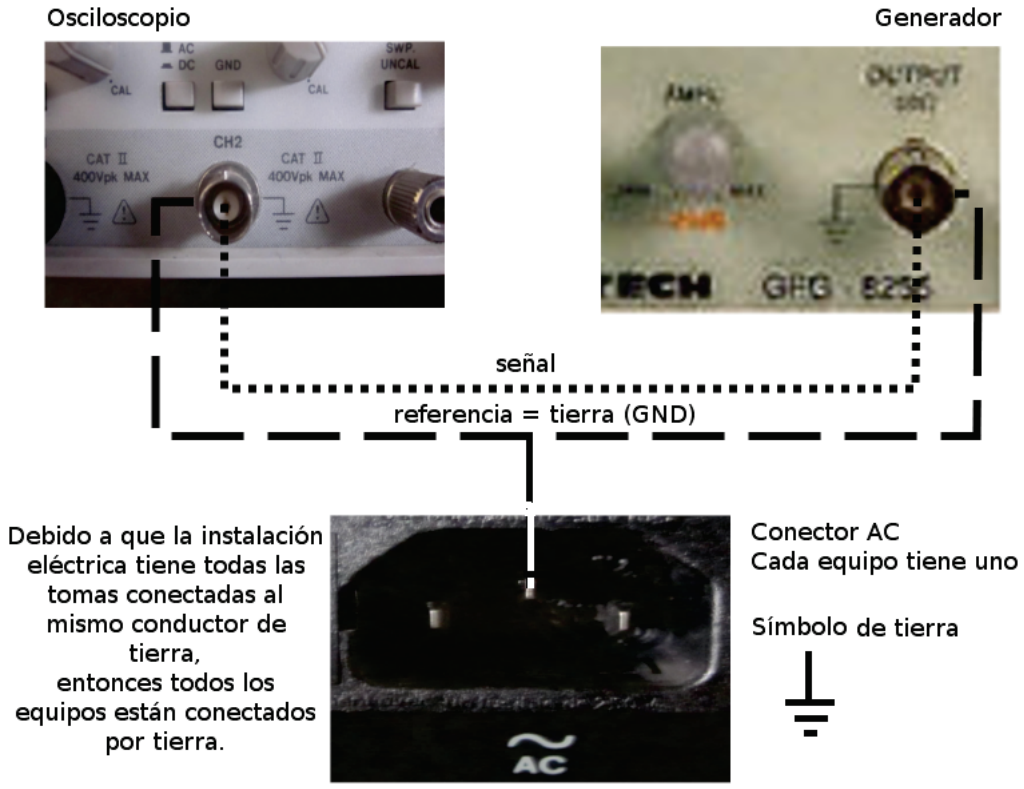


Figura B.8: Ejemplo de equipos conectados por un solo cable con tierra común.

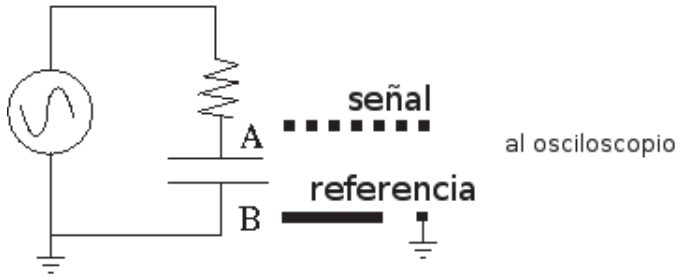


Figura B.9: Generador de funciones TEKTRONIX CFG 280.

el otro cable lo cual produciría un corto en el generador. Pero cuando hay otro equipo de por medio hay que ser cuidadoso al conectar el osciloscopio. Como ejemplo, suponga que en un circuito RC alimentado por un generador se quiere medir el voltaje a través del condensador con un osciloscopio (ver Figura B.9) . La sonda puede conectarse de 2 formas. Primero, conectando la tierra de la sonda al punto B y la parte central de la sonda al punto A. En este caso se verá efectivamente el voltaje del condensador en la pantalla del osciloscopio. Incluso, si se desconecta la sonda del punto B la señal seguirá viendose en la pantalla del osciloscopio dado que el generador y el osciloscopio tienen tierras comunes. Ahora, si la tierra de la sonda es conectada al punto A y el otro conductor de la sonda al punto B, el voltaje que el osciloscopio reportará será de 0 voltios. Lo que se ha hecho en éste último caso es poner en corto circuito el condensador.

Apéndice C

Los cubos logidule

[1]Los cubos logidule son una herramienta para comprender el funcionamiento de los circuitos combinacionales a partir de bloques funcionales básicos. Cada uno de estos bloques es un cubo, que se caracteriza por:

- Las entradas y salidas
- La función del cubo
- Las conexiones de alimentación

En la cara superior se encuentra dibujada la función que cumplen, mientras en los costados se encuentran las entradas, salidas y contactos de alimentación.

C.1. Las entradas y salidas

En ellas se realizan las conexiones que llevan las señales de un punto a otro en un sistema. Existen dos puntos de conexión:

- Los orificios en la parte superior del cubo, alrededor del simbolo que identifica su función, generalmente se encuentran hacia los bordes. En general hay uno por cada entrada y/o salida del sistema y su conexión se realiza por medio de cables que deben ser probados para evitar problemas durante la verificación del funcionamiento del circuito.[2] La

[1]Los cubos logidule fueron diseñados por la École Polytechnique Fédérale de Lausanne, Suiza [21].

[2]Ver la Sección C.4.

Figura C.1 muestra un ejemplo de conexión con cable entre la salida de dos interruptores a las entradas de una compuerta XOR, y luego la salida de la compuerta XOR a un LED.

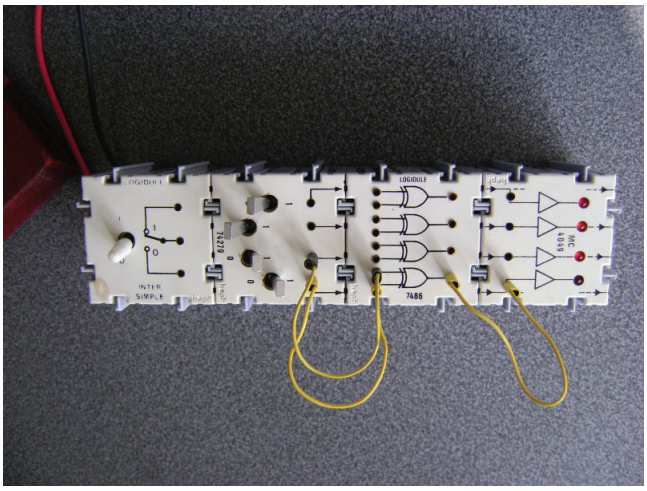


Figura C.1: Conexiones de entradas/salidas con cables.

- En los costados del cubo en la parte superior (ver Figura C.2), la parte más cercana a la cara superior. En general, no todas las entradas y/o salidas tienen punto de contacto en el costado, esto solo sucede en los casos en que el número de entradas/salidas es pequeño y hay suficiente espacio para el contacto. Sin embargo, estos contactos en los costados permiten conectar rápidamente y sin cables, las salidas de un cubo con las entradas del siguiente o la conexión en paralelo de varios dispositivos.

  Para esto, es necesario revisar cada cubo e identificar sus entradas y salidas. En general, cuando hay alguna línea que termine en el borde de la cara superior del cubo, significa que tiene una conexión lateral esa entrada/salida, en algunos casos, está señalado con flechas el sentido de flujo de las señales para identificarlas fácilmente. Cuando las líneas terminan en un semicírculo oscuro (ver Figura C.3) en el borde del cubo, se puede observar que el lado contrario también comienzan así. Esto simboliza que es la misma señal en ambos costados y pasa de un lado a otro del cubo, conectándose a la función del bloque, y además permitiendo que las mismas señales lleguen a otros bloques que se conecten

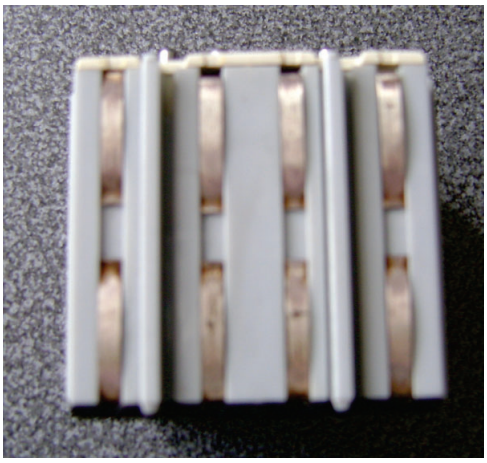


Figura C.2: Contactos laterales de los cubos para las entradas/salidas.

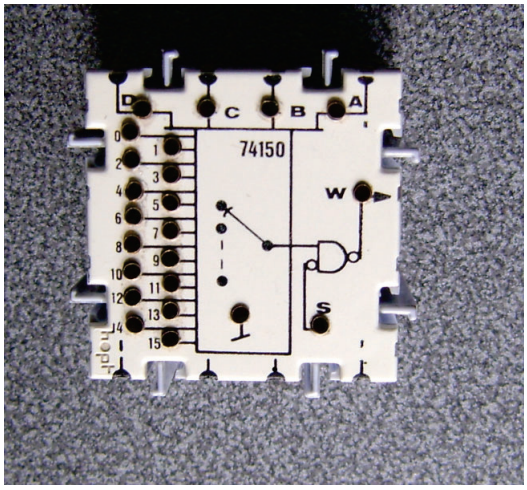


Figura C.3: Símbolo de señales que pasan de un lado a otro en un cubo.

junto a este. Este el caso de las conexiones en paralelo, se reduce el número de cables, lo cual también reduce el número de inconvenientes por fallas en las conexiones.

C.2. Las funciones

Cada cubo cumple con una función identificable por medio de símbolos colocados en la cara superior. Entre estas funciones se encuentran:

- Las compuertas lógicas
- Los buffers para encender LEDs (compuerta MC4049)
- Los valores '1' y '0' lógicos estables, latch (componente 74LS279)
- Los flip-flops JK (componente 74LS76)
- Los multiplexores (componente 74LS150)
- La fuente de voltaje (entrada de voltaje para los cubos)

La Figura C.4 muestra los símbolos correspondientes a diferentes compuertas lógicas. De izquierda a derecha y de arriba a abajo son:

- NOR de dos entradas (compuerta 74LS02)
- NAND de cuatro entradas (compuerta 74LS20)
- OR de dos entradas (compuerta 74LS32)
- NAND de dos entradas (compuerta 74LS00)
- OR de dos entradas (compuerta 74LS32)
- XOR de dos entradas (compuerta 74LS86)

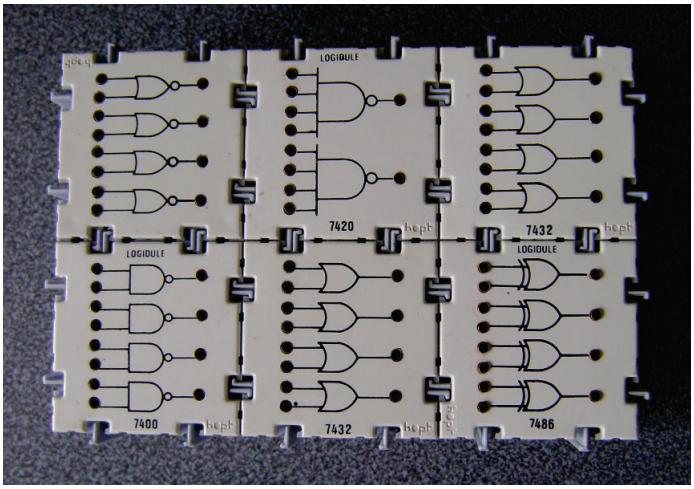


Figura C.4: Símbolos de compuertas lógicas.

C.3. Las conexiones de alimentación

Cada cubo logidule tiene conexiones para la alimentación de +5 V y GND.[3] Esta alimentación es necesaria debido a que los componentes utilizados son circuitos electrónicos activos que necesitan energía para su operación.

Los contactos correspondientes a la alimentación se encuentran en los costados en la parte inferior del cubo, tal como se muestra en la Figura C.5. En cada cara lateral están presentes para darle alimentación a los cubos que se conecten.

Nota: Coloque siempre los cubos de tal forma que coincidan las caras laterales. Las conexiones parciales o desplazadas pueden generar daños en los componentes internos en el momento de conectar la fuente de alimentación.

C.4. Pruebas preliminares

Comience con una prueba inicial para verificar el funcionamiento básico de las partes mencionadas del cubo logidule. Para esto monte el circuito mostrado en la Figura C.6 y verifique el encendido y apagado de los LEDs.

[3]La alimentación a +5 V y GND se debe al tipo de tecnología utilizada para la fabricación de los componentes electrónicos internos

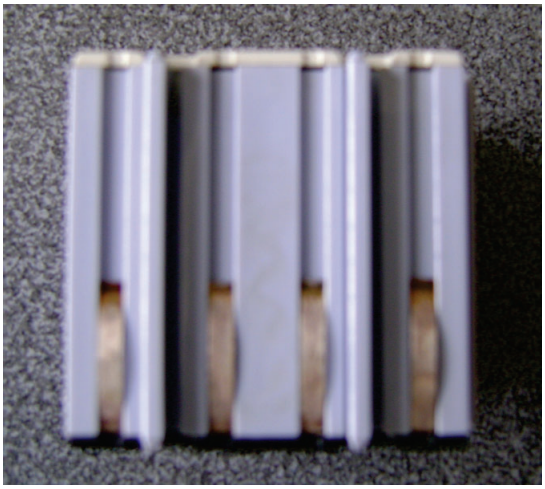


Figura C.5: Contactos laterales inferiores de alimentación.

Para esto, utilice una fuente de voltaje[4], y localice la salida de +5 VDC fijos con el fin de evitar accidentes. Colocamos el cubo inicial que corresponde a la fuente de voltaje de los demás cubos que además cuenta con un interruptor, sin embargo, este no se manipulará. Colocamos en uno de los costados, un cubo con interruptores y frente a él otro con buffers y LEDs.

Se recomienda al empezar un laboratorio en el que necesite los cubos logidule, comience montando el circuito básico mostrado en la Figura C.6. Luego rote 90º el cubo de los LEDs, el objetivo es verificar el funcionamiento de cada uno de los cables que van a ser utilizados antes de colocarlos en su posición definitiva en el montaje.[5]

En este caso, la conexión entre los cubos se hará por medio del cable que se probará, como se muestra en la FiguraC.7. Si el cable está en buen estado entonces funcionará correctamente al encender y apagar el interruptor. Pruebe moviendo un poco el cable, el LED debe permanecer estable, no debe titilar o apagarse.

[4]Consultar la sección B.1 para conocer parte de su funcionamiento.

[5]**Nota:** Al rotar el cubo de LEDs 90º, las salidas del cubo de interruptores ya no está conectada a la entrada del cubo de LEDs, por lo cual, los cables son la única unión entre estos dos puntos, tal como se explicó en la sección C.1.

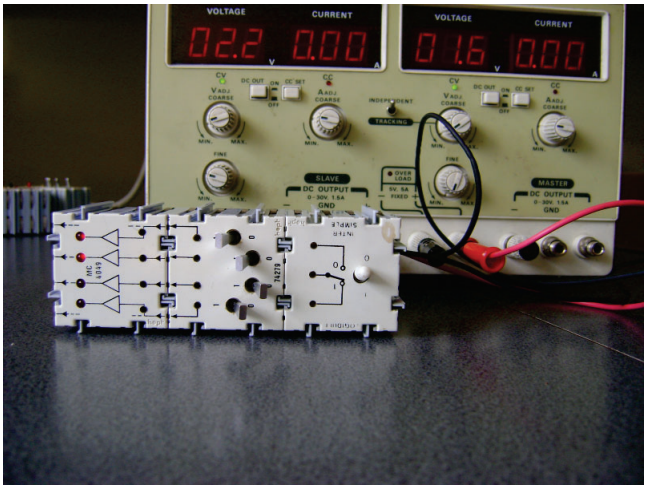


Figura C.6: Circuito básico.

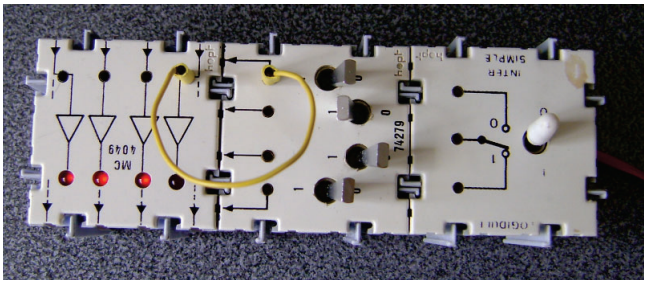


Figura C.7: Circuito básico para probar cables.

Apéndice D

El proyecto final

El proyecto final busca evaluar y poner en práctica los conceptos y habilidades aprendidas dentro del curso. Típicamente, cada grupo de estudiantes construye un sistema de medición automatizada de acuerdo a las necesidades de algún laboratorio del departamento. El proyecto final consta de una parte práctica que constituye el 23 % de la nota final y una parte escrita (ver apéndice A) que constituye el 7 % de la nota final. En los siguientes parráfos se muestran algunos ejemplos de los proyectos desarrollados en semestres pasados.

D.1. Digitalización de datos de un EPR

"El equipo de resonancia paramagnética electrónica (EPR) ha sido una herramienta eficaz y muy utilizada en el estudio y caracterización de materiales en el departemento de Física de la Universidad Nacional. Sin embargo, la manipulación de los espectros obtenidos es usualmente dispendiosa dado que estos son directamente impresos en un plotter y no es posible tener un registro electrónico de ellos en un formato estándar. Este proyecto pretende dar solución a este problema digitalizando las señales análogas del plotter y llevandolas a un computador donde los histogramas son guardados en un archivo ASCII. Para lograr este objetivo se llevan las dos señales análogas del plotter (una por cada eje) a un circuito electrónico compuesto por circuitos operacionales que convierten los niveles de voltaje entrantes al rango de 0V a 5V. Dicha señal es llevada a un microcontrolador que digitaliza las dos señales y se encarga de todo el protocolo de comunicación con el computador. En

el proceso de toma de datos con el microcontrolador se obtuvieron espectros muy aproximados a los obtenidos directamente con el plotter pero con una ligera contribución de ruido posiblemente debida a la limitada resolución de los conversores A/D que usualmente se encuentran en los microcontroladores comerciales." [22]

Este trabajo realizado por H. Camacho, J. Lamprea, S. Rodriguez, J. Cardona y O. Almanza toma las señales analógicas de salida que van normalmente hacia la impresora y las redirije hacia un circuito de acondicionamiento de señal, el cual convierte los niveles de voltaje para hacerlos adecuados a las restricciones del pic.

Los niveles de voltaje para las dos salidas son: -1 a 1 V y 0 a 10 V, y son convertidas en niveles de 0∼5 V. El circuito de conversión se realiza por medio de amplificadores operaciones LM324 alimentados a la fuente de 5 V. Las señales se llevan hasta el conversor AD del pic para digitalizar los niveles de voltaje con resolución de 8 bits. El envío de datos se inicia en el momento determinado por el computador comunicando la orden correspondiente por el puerto paralelo, luego se inicia el envío de la información del pic al pc en paquetes de 4 bits utilizando desplazamientos sobre los datos de 8 bits. Los datos son reconstruidos y almacenados en el PC para su procesamiento, donde se complementa con un post-procesamiento de la información para hallar las desviación en valor absoluto de la intensidad de la resonancia registrada.

La Figura D.1 corresponde a uno de los resultados obtenidos durante la puesta a prueba del sistema descrito, se observa una semejanza alta entre los datos enviados por el pic y los datos dibujados en la impresora. Debido a que la utilidad principal es la detección de picos o puntos discontinuos, el punto cerca a -30 u.a. es fácilmente distinguible por ambos métodos, sin embargo, el punto cercano a 4 u.a. solo se distingue correctamente en la impresión.

D.2. Automatización de mediciones de campo magnético

"Se diseñó un dispositivo que automatiza la medición de perfiles de campo magnético utilizando un posicionador mecánico motorizado sobre el cual se coloca la sonda Hall que realiza las mediciones de campo magnético a través de un gaussímetro. El proceso de medición involucra la colocación de la sonda Hall en una posición determinada a través del posicionador y un

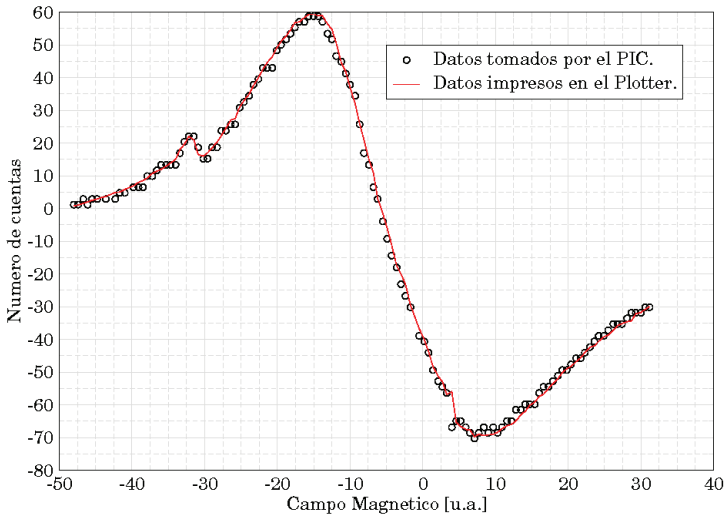


Figura D.1: Resultados de las mediciones digitales para el EPR.

computador, y la captura del voltaje proveniente del gaussímetro el cual es proporcional al campo magnético que se quiere medir. Este voltaje es enviado al conversor A/D de un microcontrolador que a su vez envía este valor digital al computador. Este proceso se repite hasta que se halla completado el recorrido total del posicionador. El resultado final es un archivo ASCII donde se pueden leer los campos magnéticos medidos para cada posición de la sonda Hall." [23]

Este trabajo realizado por W. Rodriguez, y J. Cardona consiste en el montaje mostrado en la Figura D.2. Se muestra en él un motor paso a paso unipolar que avanza 1.8º por paso. Para hacerlo mover se envía una secuencia de datos que enciende las bobinas correspondientes. La señal de la sonda de efecto Hall es llevada hasta al gaussimetro, y de éste al pic para realizar una conversión análoga-digital. El pic tiene como función la generación de la secuencia para posicionar el motor paso a paso, la conversión de valores análogicos a digitales del voltaje de salida del gaussimetro y la comunicación por puerto paralelo con el computador para el registro de los valores medidos. El computador envía las órdener de inicio de las medición y recoge los datos medidos por la sonda de efecto Hall.

La interfáz utilizada para el puerto paralelo es equivalente a la utilizada durante el desarrollo del curso. Se desarrollaron 2 programas: el primero para

el computador, encargado del control general y recepción de datos envíados por el pic, el segundo para el microcontrolador para interpretar las órdenes enviadas por el computador y ejecutarlas. Debido a esto existen diagramas de flujo separados para ambos módulos, sin embargo, ambos siguen un protocolo simple de comunicación, donde el envío de datos se hace en paquetes de 4 bits mediante desplazamientos de los valores de 1 byte de tamaño, esto duplica el tiempo de envío, sin embargo no es problemático ya que la estabilización de la medida es un proceso más lento que la transmisión para este caso.

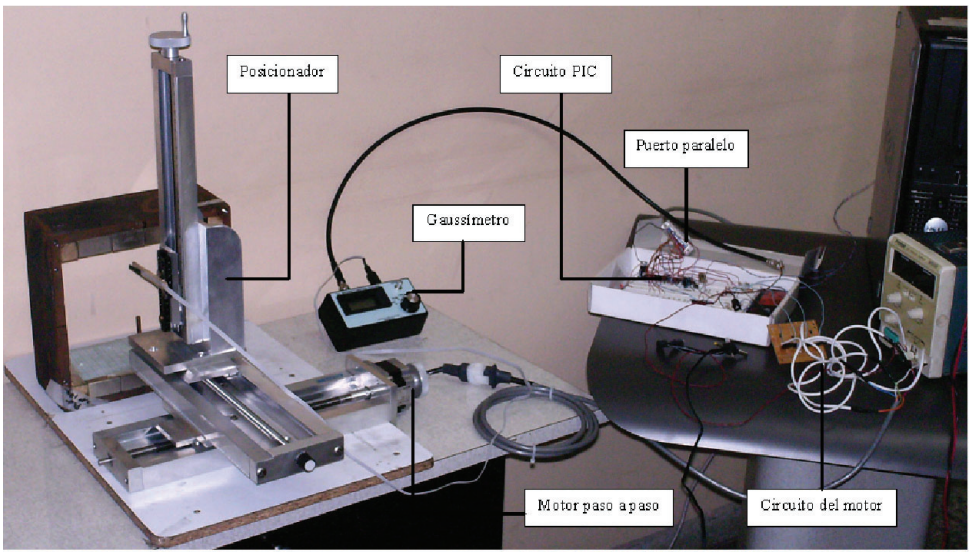


Figura D.2: Montaje realizado para la medición automatizada de campo magnético.

Apéndice E

Programación por puerto USB

E.1. General

Hasta la fecha, la programación de PICs por puerto USB se ha solucionado de la siguiente forma:

- Se escoge un pic con soporte para bootloader. El bootloader es un programa que se carga en el microcontrolador para que realice la comunicación con el PC por puerto USB.

- Se escoge un programador y se revisa que tenga soporte para puerto Serial.

- Se recurre a la información provista por microchip para cargar **una vez** el programa bootloader en la memoria del pic, por medio del puerto serial y el programador. El más utilizado localmente es el PICStart.

- En adelante, los nuevos programas se pueden cargar por puerto USB, conectando el pic al puerto directamente, de acuerdo con las especificaciones de Microchip y las características del microcontrolador.

De acuerdo a su experiencia, Vd. podrá realizar este procedimiento con los documentos guía del fabricante del microcontrolador (en este caso Microchip), sin embargo, este ya ha sido solucionado por estudiantes de la Universidad Nacional de Colombia para el PIC18F4550 en plataformas Windows, por lo cual, se recomienda leer el artículo citado en [24].

E.2. Artículo de aplicación

En este artículo [24] se utiliza un PIC18F4550 y se programa por puerto USB utilizando sistema operativo Windows.

E.2.1. Requisitos

Es necesario tener:

- Un computador con sistema operativo Windows XP (puede funcionar en otras versiones pero no se menciona).
- Programador PICStart.
- Ambiente de desarrollo PICLAB
- Simulador MPLAB SIM
- MPLAB C18 COMPILER

E.2.2. Instalación

Los siguientes son los pasos seguidos:

- Descargar e instalar MPLAB para Windows.
- Vamos al menu Configure→ Select Device y buscamos el PIC18F4550, esto confirma que el existe soporte para el pic en este programa.
- Descargamos MPLAB C18 COMPILER y lo instalamos. La Figura E.1 muestra las opciones a escoger durante la instalación del paquete.
- Después de configurar el software es necesario montar el circuito mostrado en la Figura E.2.
- Se descarga la aplicación *USB Bootloader Setup.exe*[1]. Esta aplicación trae el bootloader que permite programar el microcontrolador por puerto USB. Aún así necesitamos **una vez** grabar en la memoria esta aplicación utilizando el programador PICSTART PLUS.

[1]Esta aplicación es propiedad de Microchip Corp. y puede ser descargada del link http://ww1.microchip.com/downloads/en/DeviceDoc/USB_Bootloader_Setup.exe.

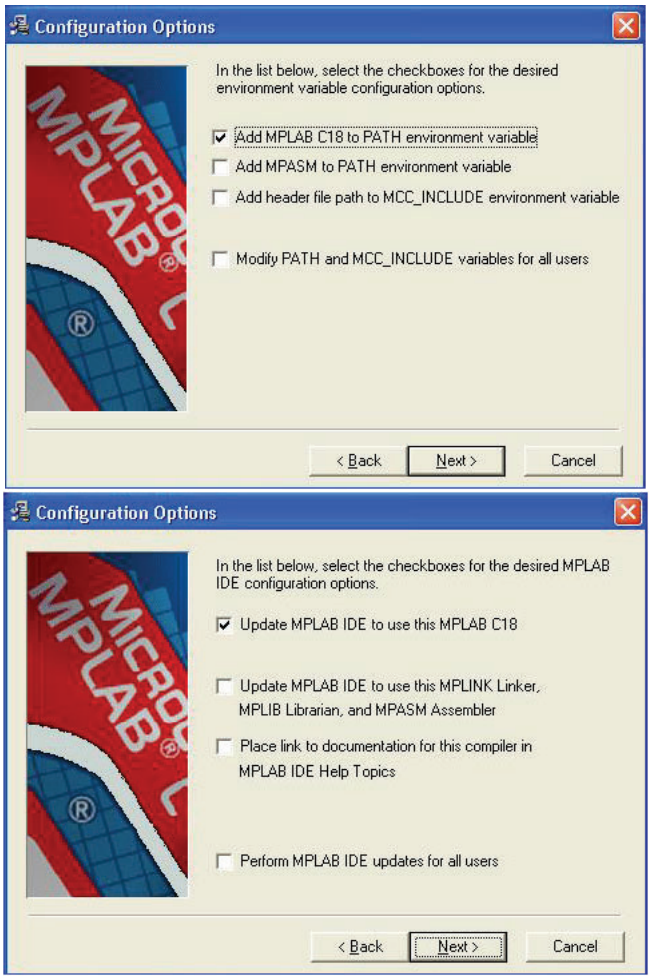


Figura E.1: Opciones de instalación de paquetes para MPLAB.

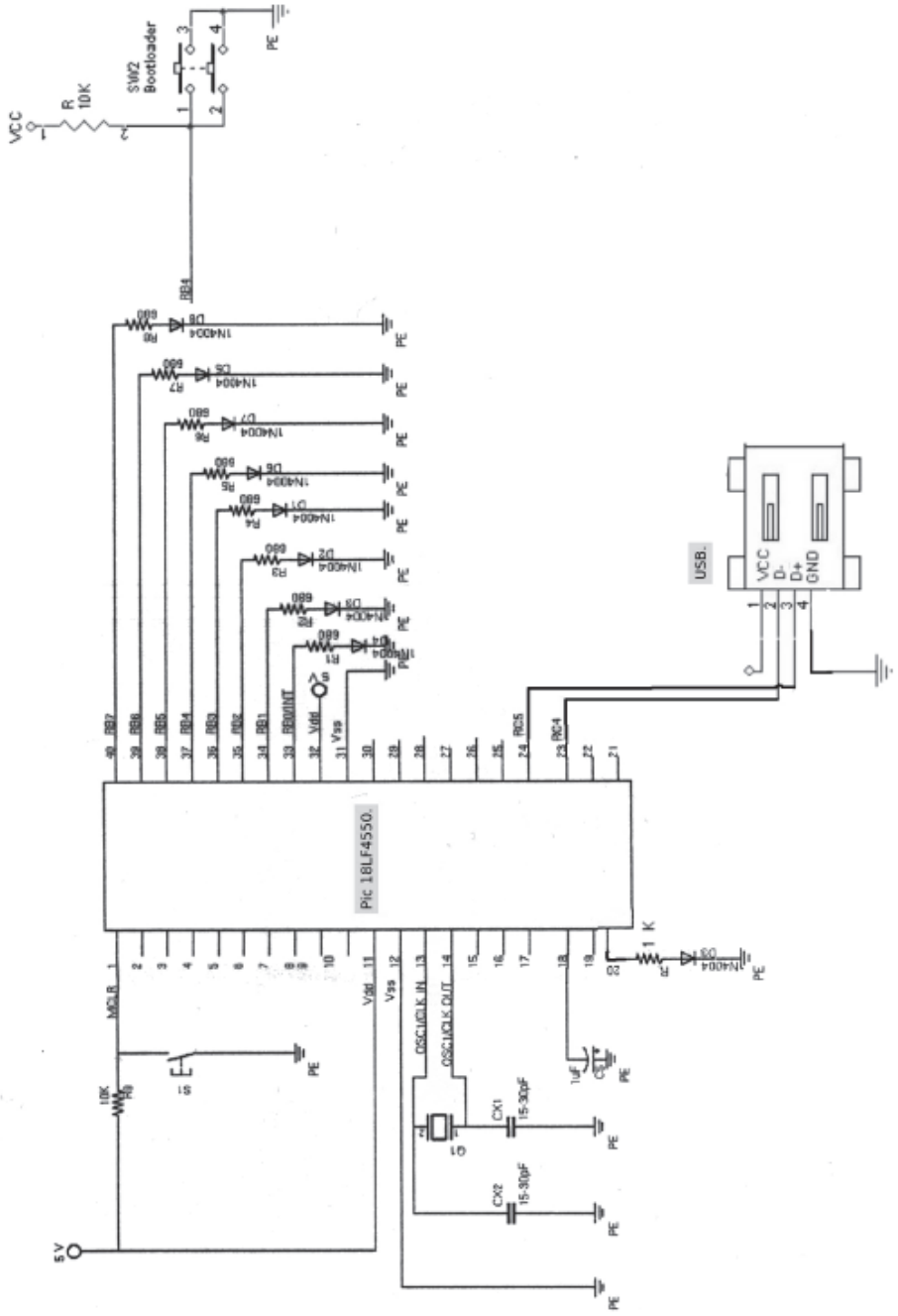


Figura E.2: Circuito para programación por puerto USB.

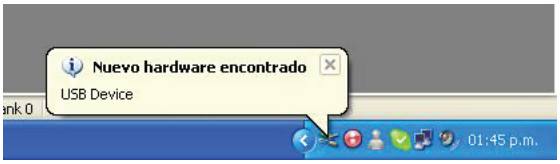


Figura E.3: Logo de detección de nuevo hardware.

Se coloca en la plataforma el pic, en MPLAB se selecciona Programmer→ Select Programmer→ PICSTART Plus, y se importa el archivo *.hex* que se encuentra en la carpeta MCHPFUSB→ fw → _factory_hex, que se genera al instalar "USB Bootloader Setup.exe".

- Se habilita el programador: Programmer → Enable Programmer y se ejecuta Program.
- Si la programación fue exitosa, aparecerá un prompt que lo confirma.
- Retiramos el pic del programador y lo colocamos de nuevo en el circuito.
- Lo conectamos al puerto USB del computador manteniendo presionado el pulsador de la línea RB4, al detectarse aparecerá el logo mostrado en la Figura E.3. y luego aparecerá la ventana mostrada en la Figura E.4, para la configuración de los drivers para el nuevo hardware encontrado.
- El driver correspondiente se muestra en la Figura E.5.
- Después de instalado el driver, cada vez que se conecte el circuito, este será reconocido automáticamente por el sistema operativo y aparecerá dentro del hardware listado en Panel de Control → Sistema → Hardware, tal como se muestra en la Figura E.6.

E.2.3. Crear un programa

Para crear un programa es recomendable seguir los siguientes pasos:

- Crear un proyecto, es preferible seguir en adelante las recomendaciones del Project Wizard.

Figura E.4: Ventana de instalación de drivers para nuevo hardware.

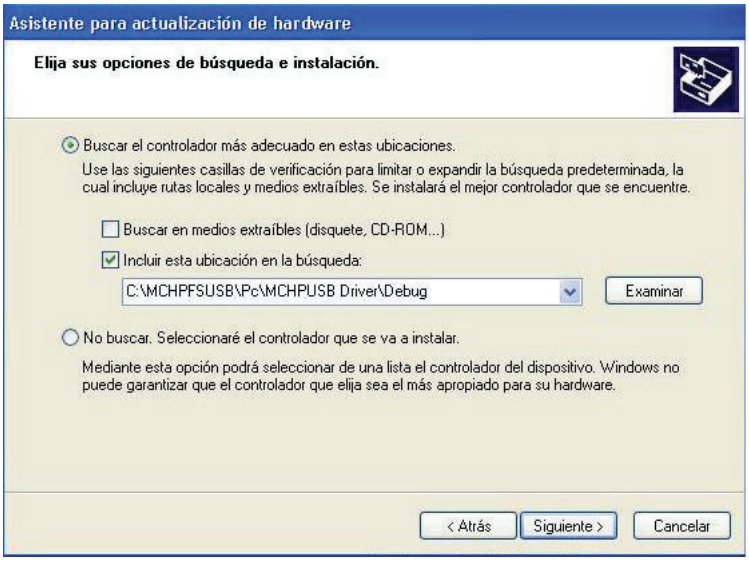


Figura E.5: Instalación de drivers para nuevo hardware.

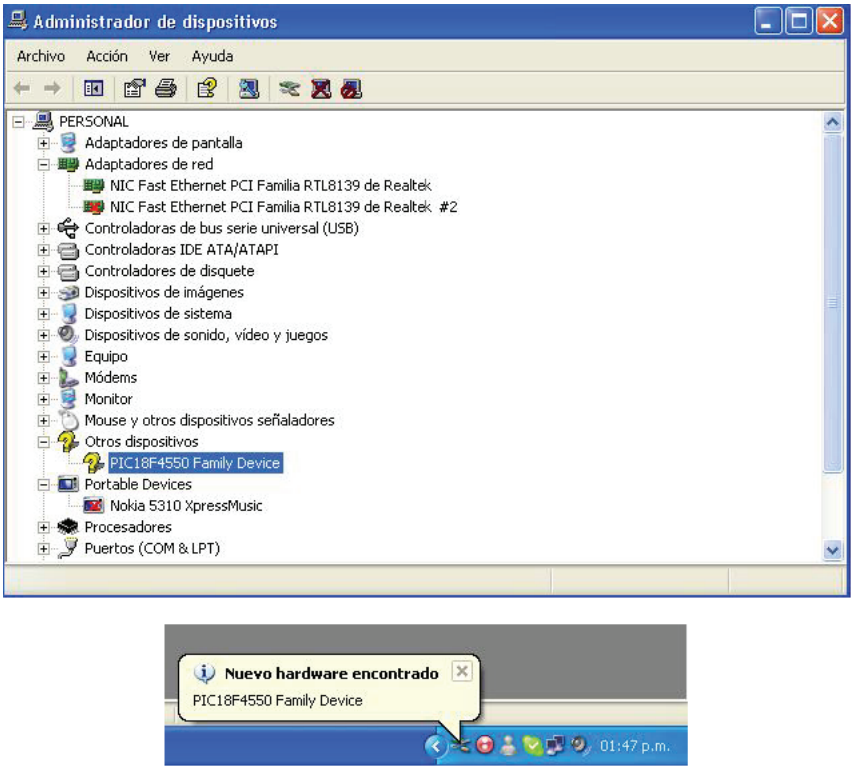


Figura E.6: Reconocimiento del hardware en el sistema operativo.

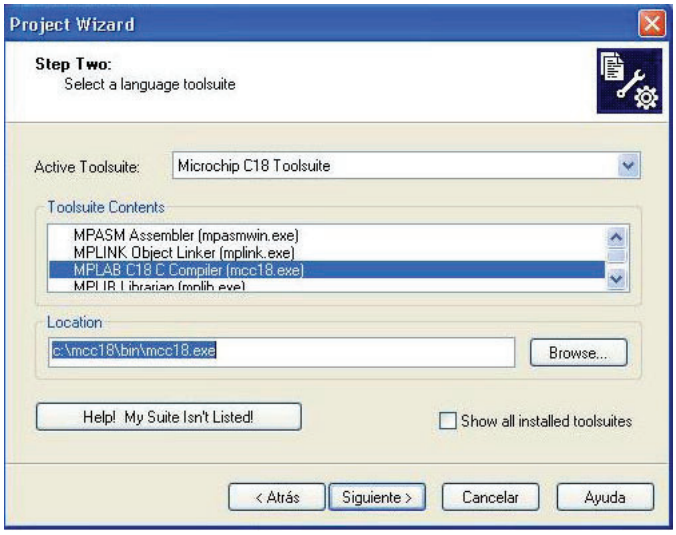


Figura E.7: Configuración del compilador C en MPLAB.

- Se elige el dispositivo a utilizar, en este caso el PIC18F4550.
- Activamos Microchip C18 Toolsuite y elegimos MPLAB C18 C Compiler (mcc18.exe), como se muestra en la Figura E.7.
- Se elige el directorio donde será guardado el proyecto.
- Aparecerá una ventana de resumen de la configuración realizada y se creará la estructura de carpetas normal para MPLAB, compuesta por:
  - Source files: donde se almacenan todos los archivos fuentes
  - Header files: para los encabezados
  - Object files: para los archivos objeto
  - Library files: para las librerías
  - Linker scripts: para el proceso de postcompilación
  - Other files: otros archivos

Ahora, vamos a File → New File para comenzar con el programa escrito en lenguage C. Para evitar sobreescribir el espacio del bootloader, es necesario desplazar las posiciones de memoria a utilizar. Esto se realiza incluyendo las siguientes líneas:

- ```
  extern void _startup (void);
  #pragma code _RESET_INTERRUPT_VECTOR = 0x000800
  void _reset (void)
  {
  _asm goto _startup _endasm
  }
  #pragma code
  ```

  además, se modifica el archivo *.lkr* para el PIC18F4550 ubicado en la carpeta MCC → lkr, añadiendo las siguientes líneas:

  ```
  CODEPAGE NAME=boot     START=0x0   END=0x7FF   PROTECTED
  CODEPAGE NAME=vectors START=0x800  END=0x829   PROTECTED
  CODEPAGE NAME=page     START=0x82A  END=0x7FFF
  ```

- Luego se añade este archivo al proyecto, haciendo click sobre la carpeta de linker files.

- Ahora, como ejemplo se presenta el código en C para encender LEDs.

  ```
  #include <p18f4550.h>
  extern void _startup (void);
  #pragma code _RESET_INTERRUPT_VECTOR = 0x000800
  void _reset (void)
    {
    _asm goto _startup _endasm
    }
  #pragma code

  void main (void)
  {
    TRISB=0X00; // todos los bits como salida
    PORTB=0xFF;
    while(1);
  }
  ```

- Guardamos este archivo .c y lo añadimos al proyecto en la carpeta *source files*. La apariencia del sistema de ventanas puede ser como se muestra en la Figura E.8
  ```

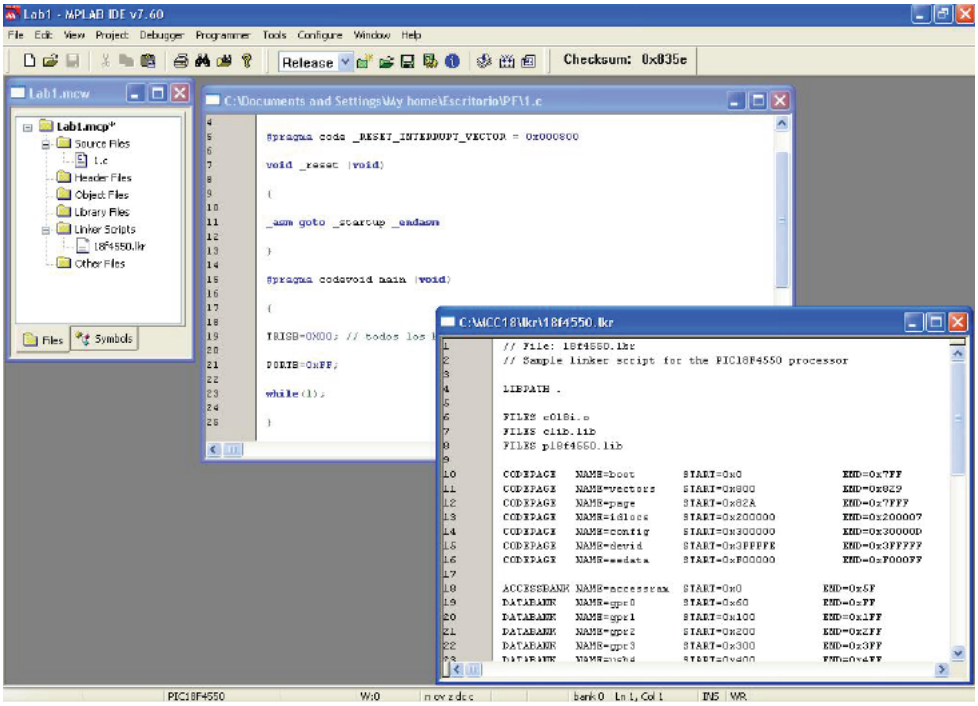


Figura E.8: Ventanas del proyecto en MPLAB.

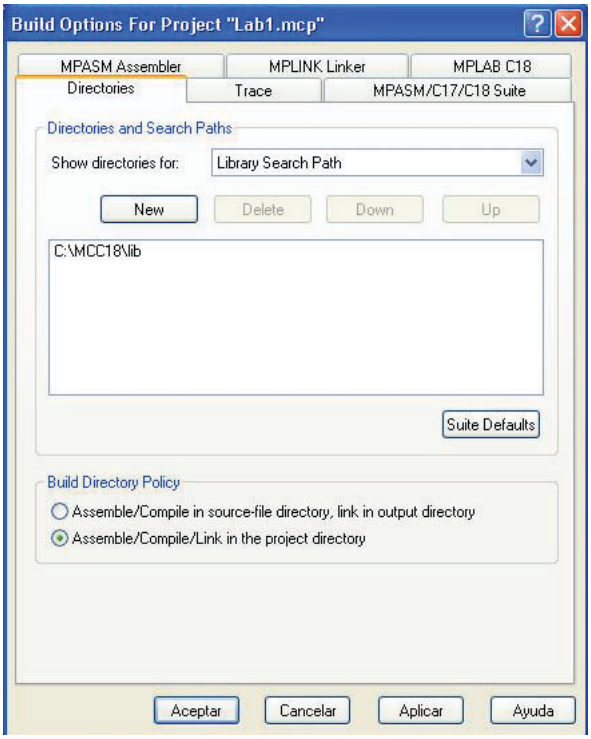


Figura E.9: Configuración de parámetros de compilación.

- Configuramos los parámetros para la compilación en Project→ Build Options → Project, de acuerdo a como se muestra en la Figura E.9.

- Se compila el proyecto en Project → Biuld All, y el archivo .hex aparecerá en la carpeta de fuentes.

E.2.4. Grabar el pic

Para grabar el pic, se abre la herramienta PIC DEM FS USB Demo Tool en la carpeta MCHPFUSB → Pc → Pdfsusb, donde aparecerá una ventana como la mostrada en la Figura E.10 Se carga el archivo *.hex* generado en la subsección anterior, y realizamos un Program y Execute[2] para transferir los datos a la memoria de pic.

[2]La función execute es equivalente a hacer un Reset del pic.

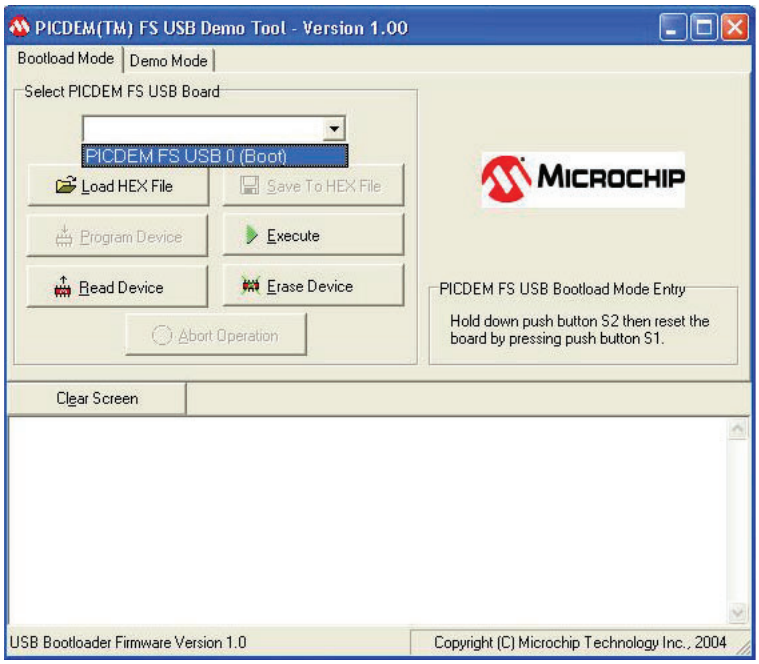


Figura E.10: Herramiento PIC DEM FS USB Demo Tool.

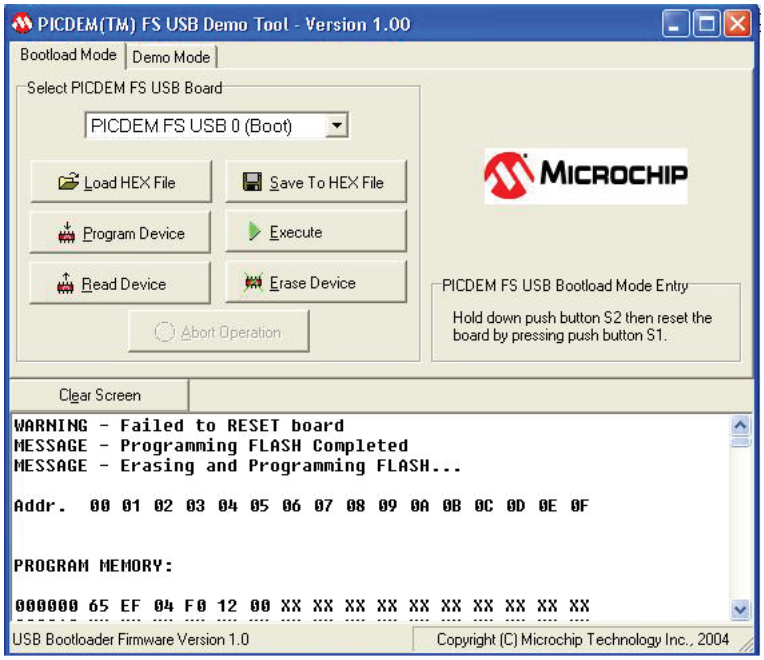


Figura E.11: Herramiento PIC DEM FS USB Demo Tool durante la programación.

La FiguraE.11 muestra la ventana durante la ejecución de la programación del pic.

E.2.5. Errores y problemas en el proceso

A lo largo de la implementación del bootloader se encuentran una serie de errores y problemas que se deben a aspectos que se pasan por alto y que son relevantes para que todo funcione correctamente.

1. Hay que asegurarse que el cristal que se usa en el circuito sea de 20 MHz, cristales de menor valor haran que una vez cargado el bootloader en el PIC, al conectarlo al computador no sea detectado, o simplemente sea detectado por unos cuantos segundos.

2. Al instalar C18 se deben habilitar las opciones mencionadas de tal manera que al trabajar con MPLAB sea posible compilar programas escritos en lenguaje c.

3. Al trabajar en MPLAB hay que especificar el PIC con el que se esta trabajando para que no se produzcan errores ya sea utilizando el simulador que trae consigo, o para el proceso de compilación.

4. Siempre hay que crear un proyecto en MPLAB para que sea posible trabajar el programa y compilarlo, no se debe olvidar añadir tanto el archivo fuente, como el linker. Para grabar los programas con el bootloader hay que tener en cuenta que es absolutamente necesario modificar el archivo linker correspondiente al PIC y añadir las lineas descritas al programa en C, de tal manera que no se ocupe el espacio de memoria ocupado por el bootloader y quede protegido.

5. Si no se especifica el camino para la busqueda de la libreria al compilar se obtendrá un error que especifica que no se encuentra el archivo c018i.o, el cual se encuentra en la carpeta M CC18 ¿lib.

6. Para que el PIC entre en modo Bootloader se debe conectar al computador presionando el pulsador que esta en el pin RB4, o simplemente presionarlo y luego presionar el otro pulsador que actua como Reset. La forma de verificar que el PIC se encuentra en modo Bootloader es viendo que el led conectado al pin 20 titila mientras esto sucede, cuando se da Reset simplemente se ejecuta el programa que este cargado en el PIC.

Apéndice F

Instalación de programas

En el directorio /usr/local/softwarepics/ de los computadores del laboratorio de instrumentación virtual se encuentran los programas que es necesario instalar antes de poder realizar las prácticas de pics de este libro. A continuación se describen cada uno de ellos y la forma como deben instalarse.

F.1. Picclite

Picclite es el compilador de C para pics y debe instalarse a partir del archivo *picclite.tar* . Los pasos a seguir para la instalación son:

1. Copiar *picclite.tar* a algún directorio de su cuenta.

2. Entrar a ese directorio y descomprimir el archivo mediante el comando `$ tar -xvf picclite.tar`

3. Desde su cuenta de usuario acceda la cuenta root por medio del comando `$ su`. Se le solicitará el pasword de root para poder entrar a esta cuenta.

4. Después de descomprimir el archivo se creará el subdirectorio *hitech/*. Copie ese subdirectorio al directorio */usr*. Para hacer esto use el comando `$ cp -r hitech/ /usr/`

5. Copie el programa *compilar* a un directorio donde el computador siempre busque cuando se llame dicho programa y dele permisos de ejecución

al programa compilar mediante el comando `chmod 770 compilar`. Para saber en que directorios el PC busca un programa, ejecute el comando `echo $PATH` y verá una lista de directorios donde el PC siempre busca.

6. Para probar la instalación salga de la cuenta de root, salga de su cuenta y vuelva a entrar a su cuenta.

7. Tome como archivo fuente cualquiera de los archivos utilizados en clase (e.g., *prueba.c*) y use el comando `compilar prueba.c`. Se deben generar los archivos *prueba.lst*, *prueba.hex* y otros archivos.

El compilador es semejante pero no igual al compilador standard de C. Por ejemplo, en C standard la declaración de una variable `char` es `signed`, mientras que para Picclite, la declaración `char` es `unsigned`. Si se requiere que tenga signo es necesario declararla explicitamente `signed char` que no es un caso típico en C.

Además, dependiendo de la forma de escritura de su programa, el compilador lo colocará en lenguaje assembler de formas diferentes, por lo cual, es recomendable utilizar las funciones que el manual del compilador recomiende.

Por esta y otras razones, tenga siempre presente que no existe una correspondencia uno a uno entre el lenguaje de máquina del pic y las instrucciones que se escriben en C.

F.2. Programas para "Quemar" el PIC

Los archivos con extensión .hex, como por ejemplo el archivo prueba.hex, constituyen el código de máquina que debe grabarse en la memoria de programa del PIC. Para realizar esta tarea existe software especializado que envia los códigos de los archivos .hex desde el computador hasta el pic a través de los puertos seriales RS-232 del mismo computador. El programa PICP es uno de estos programas y debe utilizarse junto con un programador **PICStart Plus**. La otra opción es utilizar un bootloader(como por ejemplo el Tiny PIC bootloader) que no es mas que un programa residente en la memoria del PIC que recibe los datos provenientes del computador a través de uno de sus puertos seriales sin necesidad de utilizar programador alguno. Antes de estudiar en detalle estas 2 opciones es necesario describir la forma como debe configurarse el puerto serial del computador para esta tarea.

F.3. Permisos del puerto `/dev/ttyS0`

El puerto serial utilizado para "Quemar" el PIC recibe el nombre de */dev/ttyS0* en el sistema de archivos del computador. Verifique primero cuales son los permisos actuales mediante el comando

```
$ ls -l /dev/ttyS0
```

Si el resultado es un usuario o un grupo al cual usted tenga los permisos de leer y escribir, no es necesario cambiar nada.

Si no es así, para habilitar la lectura y escritura sobre el puerto serial, ejecute:

```
# chown <su-usuario>:<su-grupo> /dev/ttyS0
# chmode 770 /dev/ttyS0
```

Probablemente estos pasos tenga que ejecutarlos cada vez que inicie el computador. Si desea que estos cambios sean permanentes y tiene algo de experiencia manejando linux puede tratar lo siguiente:

1. Dirijase a */etc/udev/rules.d*
2. Determine que archivo coloca los permisos sobre el puerto */dev/ttyS0*. **Sugerencia:** Busque entradas `tty`, ya que estos archivos suelen configurar los permisos para todas las terminales a la vez. Puede aparecer como `tty[A-Z][0-9]`
3. Edite el grupo, el dueño y los permisos correspondientes

Si lo prefiere, también puede crear una regla adicional y adicionarla al sistema que solo haga lo anterior y sea el último en ejecutarse. Este es un procedimiento más confiable ya que el archivo puede ser eliminado sin causar problemas en el sistema.

F.4. PICP

Picp es un programa utilizado como interfaz en Linux desde consola para el programador **PICStart Plus**. El Picp se puede conseguir en [25] y una vez instalado se puede invocar a través de una consola especificando el puerto serial de salida, el PIC, el número de palabras y el archivo a grabar en el PIC. Por ejemplo, el archivo prueba.hex se graba a través del comando:

```
$ picp /dev/ttyS0 16f877 -s100 -wp prueba.hex
```

Realmente no es necesario que instale este programa ya que en el laboratorio se utiliza solo una vez durante todo el semestre.

F.5. PIC Bootloader

El bootloader permite grabar un programa en la memoria del PIC sin necesidad de sacar el PIC del protoboard a diferencia de lo que sucede con los programadores convencionales. Esto trae como ventaja una disminución significativa del tiempo de desarrollo y de los riesgos de daño en el PIC y sus circuitos anexos. Existen una gran variedad de bootloaders disponibles para PICs del cual hemos escogido el Tiny PIC bootloader que puede descargarse libremente de
`http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm`.
Para grabar este bootloader en la memoria del PIC es necesario usar un programador como el PICSTART Plus y el software PICP. Esto usualmente se hace en la primera práctica de PICs del curso en el laboratorio de instrumentación virtual mediante el comando:
`$ picp /dev/ttyS0 16f877 -s100 -wp tinybld16F.HEX` y si todo va bien no es necesario volver a hacerlo durante todo el semestre. De ahí en adelante la grabación de los archivos .hex se hace directamente en el circuito final del pic y un software particular que envía los archivos .hex al puerto RS-232 del computador. Para el caso del Tiny PIC bootloader, este software está escrito solo para el sistema operativo Windows. Es posible, sin embargo, encontrar software libre en la red de desarrolladores independientes, como por ejemplo el pytbl.py (`http://www.cihologramas.com`) escrito en python y usando la librería python-serial para correr en Linux.

F.6. Python-Serial

Este programa se encarga de la comunicación del computador con el pic a través del puerto serial RS232. La gran mayoría de las distribuciones de Linux tiene python instalado por defecto. Sin embargo, no sucede lo mismo para python-serial. Para saber si python-serial esta instalado vaya al directorio */usr/lib*. Una vez en este directorio escriba el comando `ls -d pyt*/`. Aparecerá el nombre del directorio python que está instalado. Vaya a ese directorio y luego vaya al directorio *site-packages*. Estando en el directorio

site-packages escriba el comando `ls -d */`. Si aparece el directorio *serial/*, el python-serial ya está instalado. Si no es así, copie el directorio serial que se encuentra en el directorio *softwarepics/serial* al directorio donde copió el programa *compilar*.

F.7. Pytbl.py

PyTBL es el acrónimo de *Python interface for the Tiny PIC Bootloader*, fue desarrollado por el profesor Ricardo Amézquita Orozco, propiedad de Combustión Ingenieros Ltda. `http://www.cihologramas.com`, y distribuido bajo la licencia GNU. Este programa es el que lee el programa fuente en formato *hex* y lo graba en la memoria del pic. Para instalarlo simplemente copie los archivos *pytbl.py* y *quemar* al mismo directorio donde copió el archivo *compilar*

Vaya al directorio donde copió el archivo y dé permisos de ejecución mediante el comando `$ chmod 550 pytbl.py` y `$ chmod 770 quemar`

Para probar la instalación, python-serial debe estar debidamente instalado y los permisos del puerto deben estar debidamente configurados tal como se explicó en las secciones anteriores. Salga de la cuenta de root, salga de su cuenta y vuelva a entrar a ella. Localice un archivo compilado (e.g. *prueba.hex*) y ejecute el comando `$ quemar prueba.hex` tal como se explica en la práctica 6

La versión más reciente de este programa es publicada para su descarga en `http://code.google.com/p/pytbl/`, la cual es la página del proyecto.

F.8. Conversor USB-RS232

Existe la posibilidad de realizar la programación del microcontrolador usando el puerto USB. Para ello es necesario adquirir un dispositivo electrónico conocido como **Conversor USB-RS232**. Este conversor realiza la interfaz entre el protocolo de comunicación USB y el protocolo de comunicación serial. Se distingue fácilmente debido a que en general, tiene un conector USB en un extremo, y un conector serial DB9 con una cubierta de gran tamaño comparada con un conector serial convencional debido a que allí está la electrónica que realiza la conversión de los protocolos en ambas direcciones.

Su instalación puede realizarse fácilmente en cualquier sistema operativo Windows o Linux. Si se requiere trabajar en Windows, es necesario que el fabricante del Conversor incluya los drivers adecuados. Si se trabaja en Linux, el kernel tiene automáticamente el soporte para estos conversores a través del módulo **usbserial** que debe ser instalado o habilitado (en muchas instalaciones ya esto es hecho por defecto).

Una vez se el sistema operativo reconoce el nuevo dispositivo conectado, automáticamente crea una entrada en la lista de dispositivos con el nombre */dev/ttyUSB0*. Si se coloca más de un conversor, el sistema los enumera consecutivamente. Esta entrada */dev/ttyUSB0* funciona como un dispositivo virtual. Sobre él se pueden ejecutar todas las funciones de programación correspondientes a un puerto serial real (lectura, escritura, configuración, etc.), sin embargo, el sistema se encargará de direccionar la información hacia el puerto USB donde fue conectado el conversor.

A partir de este punto, se puede comunicar con el pic a través del puerto */dev/ttyUSB0* como si fuera un puerto serial común, con el conector USB conectado al PC y el extremo serial a la entrada serial del montaje de la Figura 6.1.

Si su computador solo cuenta con puertos USB y ningún puerto serial, entonces puede adquirir permisos de root y cambiar el nombre del puerto de */dev/ttyUSB0* a */dev/ttyS0*. Con esto no es necesario cambiar nada más en los archivos.

En caso que el computador tuviese puertos USB y puertos seriales, entonces, con el fin de evitar conflictos con el hardware, los programas que se realicen en el PC ahora deben editarse para abrir el puerto */dev/ttyUSB0* y **no** el puerto */dev/ttyS0* que por defecto es el primer dispositivo serial. Además, en el momento de ejecutar `pytbl.py`, para programar el dispositivo, es necesario que se cambié de puerto destino. Así, para programar el archivo *prueba.hex* en el PIC usando el conversor USB-RS232, el comando a utilizar será:

```
$ pytbl.py -f prueba.hex -p /dev/ttyUSB0 -b 19200
```

Por favor verifique que el pytbl.py instalado tiene soporte para colocar la ruta del puerto. Versiones iniciales de este programa solo permitían la inclusión de un número de cada puerto. Se recomienda adquirir la versión más reciente de este programa como se menciona en la sección F.7 o cambiar de nombre el puerto automáticamente generado como se mencionó anteriormente.

Apéndice G

Librerías para el manejo del protocolo RS-232

La librería de comunicación que se usa para las prácticas descritas en este texto se encuentra en el directorio /usr/local/softwarepics/rs232-v6/ de la sala de instrumentación virtual o bién en el CD anexo. Dentro de la carpeta rs232-v6/ existe el archivo *README.txt* que describe el contenido y la forma de uso de cada parte en forma detallada.

En esta carpeta se encuentran archivos para el PC, el PIC y archivos de ayuda:

- adpic.c, adpc.c: Establecen la comunicación entre el PIC y el PC correspondientemente, realiza conversiones del AD y las envía por el puerto serial.
- asciifull.gif: Este archivo es una imagen del codigo ASCII para que sirva como guia durante la codificación y las pruebas. Tomado de http://www.asciitable.com/.
- **capser.***: Definición de las funciones especiales utilizadas por el PC y creadas por los autores de este README + la colaboración publicada en internet de

```
ser.c.
(C) 2004-5 Captain http://www.captain.at
Sends 3 characters (ABC) via the serial port (/dev/ttyS0)
and reads them back if they are returned from the PIC.
```

```
Used for testing the PIC-MMC test-board
http://www.captain.at/electronic-index.php
```

Son de resaltar las funciones:

- `initport(int fd)`: Esta función establece los parámetros de configuración del módulo de transmisión serial del PC. La configuración corresponde a 9600 baudios, sin control de flujo, sin bit de paridad, un bit de stop, 8 bits de datos, sin tiempo de retardo, lectura cada que llegue un nuevo dato. Retorna 1 al terminar correctamente.
- `abrirpuerto_ttyS0()`: En linux, los puertos se manejan como archivos, por lo esta función abre un archivo en la ruta */dev/ttyS0*, retorna el apuntador al archivo si termina correctamente o -1 si existe algún problema.

- lab_ad.*: Contiene una copia del capítulo 8.

- **Makefilepc, Makefilepic**: En estos archivos está construida la secuencia de compilación para el código fuente del PC y del PIC. Revise su contenido para familiarizarse con su funcionamiento.

  La forma de utilizarlo para el PC es mediante el comando:

  ```
  $ make -B -f Makefilepc PROGRAMA=programapc
  ```

  donde *programapc* es el nombre del archivo fuente que se va a compilar para el PC.

  La forma de utilizarlo para el PIC es mediante el comando:

  ```
  $ make -B -f Makefilepic PROGRAMA=programapic
  ```

  donde *programapic* es el nombre del archivo fuente que se va a compilar para el PIC.

- README.txt: Contiene una descripción rápida de la utilización de los archivos makefile.

- **rs232.***: Librerías de funciones para la comunicación serial por UART. Son de resaltar las funciones:

- `sender(x[ ],n)`: Envía `n` bytes del arreglo `x[ ]` por el puerto serial. Toma como parámetro la dirección de una arreglo cuyos datos son de tamaño `char` y repite el ciclo de envío uno por uno cargando los datos en la UART y esperando a que sea enviado para continuar con el siguiente. Nótese que si quiere enviar un solo dato, es necesario pasar el argumentos referenciado. e.g. si tengo un variable `char dato` y quiero enviarla, la declaración correcta es `sender(&dato,1)`.
- `receiver(x[ ],n)`: Espera hasta recibir `n` bytes en el arreglo `x[ ]`. El hardware de la UART recibe bit por bit y cuando el byte 'i' es recibido completo, es enviado a la posición de memoria de `x[i]` (de tamaño `char`) hasta completar los `n` bytes. ¡La función espera hasta completar los `n` bytes!. Nótese que si se quiere recibir un byte en una variable `char` y no en un arreglo, entonces es necesario pasar el argumento referenciando. e.g. si tengo la variable `char dato` y quiero recibir en ella, la declaración correcta es `receiver(&dato,1)`.
- `config_UART()`: Determina los parámetros de configuración del módulo USART del pic para que funcione en modo ASINCRONICO FULL DUPLEX, a 9600 baudios, 1 bit de stop, 8 bits de datos.

- tx1pc.c, tx1pic.c: Estos dos programas hacen que el PC y el pic envíen y reciban datos uno tras del otro de a un byte.
- tx2pc.cm, tx2pic.c: Estos dos programas hacen que el PC y el pic envíen y reciban datos uno tras del otro de a dos bytes.

No se especifican funciones especiales de escritura y lectura del puerto debido a que en Linux, los puertos se manejan como archivos, por lo cual, las funciones standard `read()` y `write()` de C son suficientes. Dentro de los ejemplos se encuentra mostrada su implementación.

Bibliografía

[1] Albert Paul Malvino. *Principios de electrónica*. Mc Graw Hill, septima edición edition, 2008. Traducción al español del libro Electronic Principles.

[2] Philips Semiconductors. *2N2222, 2N2222A NPN Switching transistors*, may 1997. Hoja de datos de la compañia NPX.

[3] General Electric. *Transistor Manual*. General Electric, quinta edición edition, 1960.

[4] Robert Boylestad Louis Nashelsky. *Electrónica: Teoría de Circuitos*. Pearson Education, sexta edición edition, 1997. Traducción al español del libro Electronic Devices and Circuit Theory.

[5] Intersil Inc. *LM324, Quad, 1MHz, Operational Amplifiers for Commercial, Industrial and Military Applications*, 796.5 edition, may 2001. Hoja de datos de componente obsoleto.

[6] Richard Dorf y James Svoboda. *Circuitos eléctricos.* Alfaomega, sexta edición edition, 2006.

[7] Morris Mano. *Lógica Digital y Diseño de Computadores*. Prentice Hall, primera edición edition, 1982. Traducción al español del libro Digital Logic and Computer Design.

[8] Motorola Inc. *1-OF-8 DECODER/MULTIPLEXER*, rev. 5 edition. Hoja de datos sn74ls138rev5.pdf.

[9] Fairchild Semiconductor Corp. *DM74LS20 Dual 4-Input NAND Gate*, mar 2000. Hoja de datos 375505_DS.pdf.

[10] Faichild Semiconductor Corp. *3-Terminal 1A Positive Voltage Regulator*, 2001. Hoja de datos de componente 390068_DS.pdf.

[11] R.J. Tocci, N.S. Widmer, and G.L. Moss. *Digital systems: principles and applications*. Number n.º 1 in Digital Systems: Principles and Applications. Pearson/Prentice Hall, 2004.

[12] Microchip Technology Inc., http://www.microchip.com/. *PIC16F87X Datasheet 28/40 Pin 8-Bit CMOS FLASH Microcontrollers*, ds30292c edition, 2001.

[13] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall, segunda edición edition, 1988.

[14] HIGH-TECH Software, http://www.htsoft.com/. *PICLITE C Manual*, primera edición edition, 2000.

[15] Texas Instruments Inc. *MAX232, MAX232I Dual BA-232 DRIVERS/RECEIVERS*, 2002. Hoja de datos max232.pdf.

[16] D.D. Gajski. *Principles of digital design*. Prentice Hall, 1997.

[17] Raymond E. Frey. Lecture notes for digital electronics. marzo 2002.

[18] Hi-Tech Software, http://www.htsoft.com/. *PICC Lite ANSI C Compiler User's Guide*, seventh edition, 2004.

[19] ScienceProg. Prototype board types. http://www.scienceprog.com/prototype-board-types/, 2006. Acceso: Enero 20 de 2011.

[20] Materia y Movimiento. Arma tu arduino. http://taller.tagabot.org/index.php/Arduino/Armada, 2010. Acceso: Enero 20 de 2011.

[21] École Polytechnique Fédérale de Lausanne. Logidule simulator. http://diwww.epfl.ch/w3lami/logidules.html, 1999. Acceso: Enero 24 de 2010.

[22] H. Camacho J. Lamprea S. Rodriguez O. Almanza J. Cardona. Digitalización de datos para un espectrometro de resonancia paramagnética electrónica. *Congreso Nacional de Física, Universidad de Magdalena, Santa Marta, Colombia*, XXIII(IM-OR-04):177–178, 2009.

[23] W. Rodríguez J. Cardona. Automatización de la medición de perfiles de campo magnético en imanes y electroimanes utilizados en aceleradores de partículas. *Congreso Nacional de Física, Universidad de Magdalena, Santa Marta, Colombia*, XXIII(IM-OR-26):191, 2009.

[24] M. P. Rozo A. Acevedo. Bootloader con usb en windows para pic18f4550. Trabajo final de electrónica digital, dec 2008.

[25] Jeffery L. Post. Development tools for pic programmers. http://home.pacbell.net/theposts/picmicro/, 2008. Acceso: Enero 6 de 2010.

Índice alfabético